1/5/1 **(Item 1 from file: 351)**
DIALOG(R)File 351:Derwent WPI

012903002 **Image available**
WPI Acc No: 2000-074838/200007
XRPX Acc No: N00-058713

Distributed software system for computer network

Patent Assignee: NEC CORP (NIDE); NEC RES INST INC (NIDE)
Inventor: FUJITA S; JAGANNATHAN S; KELSEY R A; KOYAMA K; PHILBIN J F; YAMANOUCHI T

Number of Countries: 027 Number of Patents: 003
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
EP 969364	A2	20000105	EP 99111370	A	19990610	200007 B
JP 2000020487	A	20000121	JP 9947025	A	19990224	200015
US 6496871	B1	20021217	US 98109412	A	19980630	200307

Priority Applications (No Type Date): US 98109412 A 19980630
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
EP 969364	A2	E	50	G06F-009/46	
	Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT LI LT LU LV MC MK NL PT RO SE SI				
JP 2000020487	A		41	G06F-015/16	
US 6496871	B1			G06F-009/54	

Abstract (Basic): EP 969364 A2

NOVELTY - A first object on a first base may access a second object on a second base without knowledge of the physical address of the second object on the second base. At least one runtime system is connected to the first base and the second base. The runtime system facilitates migration of agents and objects from at least the first base to at least the second base.

DETAILED DESCRIPTION - When an object (55) is created in the Agent on Base (1) and Machine (1), its identity and location are recorded in Agent's object space (70), together with a symbolic (or remote) reference (55') to object (55). Subsequent references to object (55) may be made using symbolic reference (55'), which are handled by the system by querying the object space (70) and resolving the symbolic reference (55') to the appropriate physical location for object (55). Thus, object space (70) enables transparent access to elements in Agent, which spans across multiple physical machines.

An INDEPENDENT CLAIM is included for:

(a) a method for implementing a network-centric computer software programming system for a multi-computer network

USE - In the field of distributed and parallel computer software programming, and in particular to a software system including distributed agents which exhibits enhanced process mobility and communication and facilitates the construction of network-centric applications suited for both homogeneous and heterogeneous environments.

ADVANTAGE - Easy to program and provides an object-based encapsulation model, such as an agent, which allows the processes and state of the agent to be distributed over multiple potentially heterogeneous machines, enables transparent access of data resident on another machine and allows easy and efficient process migration, in whole or in part, among distinct machines

DESCRIPTION OF DRAWING(S) - The drawing is a conceptual diagram showing the operation of an object space in distributed agent system according to the present invention.

object (55)

Agent's object space (70)

pp; 50 DwgNo 3/17

Title Terms: DISTRIBUTE; SOFTWARE; SYSTEM; COMPUTER; NETWORK
Derwent Class: T01

International Patent Class (Main): G06F-009/46; G06F-009/54; G06F-015/16
International Patent Class (Additional): G06F-009/44
File Segment: EPI

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－20487
（P2000－20487A）
(43)公開日　平成12年１月21日(2000.1.21)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｆ　15/16	６２０	Ｇ０６Ｆ　15/16	６２０Ｗ	
9/44	５５２	9/44	５５２	
9/46	３６０	9/46	３６０Ｂ	

審査請求　有　　請求項の数57　ＯＬ　（全 41 頁）

(21)出願番号　特願平11－47025

(22)出願日　平成11年２月24日(1999.2.24)

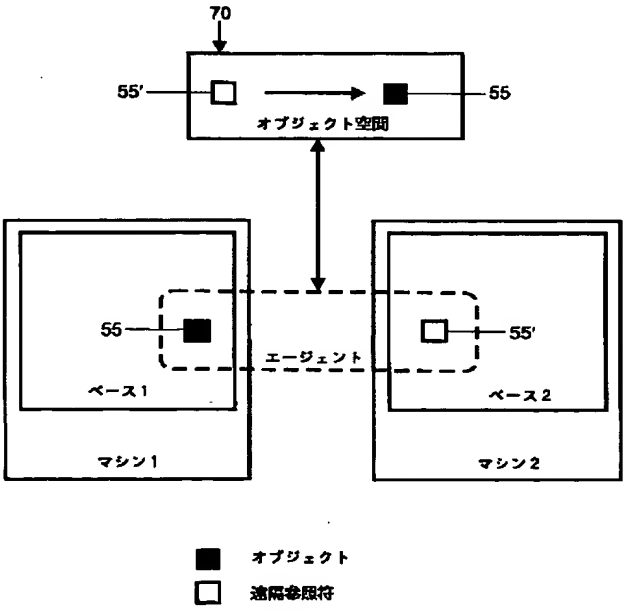
(31)優先権主張番号　０９／１０９４１２
(32)優先日　平成10年６月30日(1998.6.30)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(72)発明者　スレッシュ　ジャガナサン
アメリカ合衆国、　ニュージャージー　08540、　プリンストン、　インディペンデンス　ウェイ　４　エヌ・イー・シー・リサーチ・インスティテューテュ・インク内
(74)代理人　100070219
弁理士　若林　忠　（外４名）

最終頁に続く

(54)【発明の名称】　ネットワークとして接続された複数のコンピュータマシン用の分散ソフトウェアシステム及びその実現手法

(57)【要約】
【課題】　エージェントがそれのタスクおよび状態を、ネットワーク内で複数の異質であってもよい物理的マシン間に分散させることができる分散ソフトウェアシステム及びその実現手法を提供する。
【解決手段】　単一のエージェントであるエージェントが、マシン１およびマシン２という２つのマシン上のベース１およびベース２という２つのベースに広がっている。オブジェクト５５がベース１およびマシン１のエージェントで作られると、それの識別名称および位置がオブジェクト５５に対する記号（または遠隔）参照符５５’とともに、エージェントのオブジェクト空間７０に記録される。オブジェクト５５に対するその後の参照は記号参照符５５’を用いて行うことができ、システムはそれをオブジェクト空間７０に問い合わせオブジェクト５５についての該当する物理的位置に対する記号参照符５５’を解くことで処理する。



1

【特許請求の範囲】

【請求項１】　複数のベースがあり、前記ベースの各々が、複数のコンピュータマシンのうちの１台上で、ローカルアドレス空間およびコンピュータリソースを提供する事を特徴とし、１以上のエージェントがあり、前記エージェントの各々が保護領域を構成し、該１以上のエージェントの前記保護領域が前記複数のベースのうちの１以上のベースにある事を特徴とし、

前記１以上のエージェントの前記保護領域内に複数のオブジェクトがあり、第１のオブジェクトが前記複数のベース中の第１のベースにあり、第２のオブジェクトが前記複数のベース中の第２のベースにあり、前記第１のベースにある前記第１のオブジェクトが、前記第２のベースにある前記第２のオブジェクトの物理アドレスがわからなくとも、前記第２のベースにある前記第２のオブジェクトにアクセスすることができる事を特徴とし前記第１のベースおよび前記第２のベースに関連する１以上の実行時システムであって、少なくとも前記第１のベースから少なくとも前記第２のベースまでのエージェントおよびオブジェクトの移動を容易にする実行時システムとから構成される、ネットワークとして接続された複数のコンピュータマシンで用いるための分散ソフトウェアシステム。

【請求項２】　各エージェントは、さらに、１以上のサブエージェントから構成され、

該各サブエージェントは、１つのベースにあり、

該サブエージェント中のオブジェクトを保存するオブジェクトメモリーと、

該サブエージェント中のタスクフレームを保存するタスクメモリーと、

該サブエージェントが属する前記エージェントのためのプログラムコードと、

サブエージェント制御ストレージとから構成され、

該サブエージェント制御ストレージは、

前記サブエージェントが属する前記エージェントを示すエージェントＩＤと、

オブジェクトの記号参照符を前記オブジェクトメモリー中で前記オブジェクトの対応する物理アドレスにマップするマッピングを有するオブジェクトテーブルと、

前記タスクメモリー中に複数のタスクスレッドポインタを保存するタスクスタックとから構成され、

前記１以上の実行時システムは、さらに、

対応するベースにあるサブエージェントの、複数のサブエージェント制御ストレージを管理する、各ベース用のエージェントマネージャと、

対応するベースにある複数のサブエージェントのための、複数のオブジェクトメモリーを管理する、各ベース用のオブジェクトマネージャと、

前記対応するベース以外の１以上のベースに、前記ネットワークによって前記オブジェクトを伝送するための、

2

前記オブジェクトを直列化する各ベース用のオブジェクト直列化器と、

プログラムコードを読み、タスクメモリーにタスクスタックを作り、前記プログラムコードを実行する、各ベース用のタスクエグゼクタと、

前記対応するベース以外の１以上のベースに、前記ネットワークによって前記タスクスタックを伝送するための、前記タスクスタックを直列化する各ベース用のタスク直列化器と、

前記対応するベース以外の１以上のベースでタスクエグゼクタから遠隔オブジェクトアクセスメッセージを受け取り、前記対応するベース以外の１以上のベースに対して遠隔オブジェクトアクセス要求を送る、各ベース用の遠隔アクセスコントローラと、

前記コンピュータマシンと他の前記コンピュータマシンとの間の物理的通信を調整する通信システムとから構成される請求項１に記載の分散ソフトウェアシステム。

【請求項３】　前記プログラムコードは、前記サブエージェント内にクラスファイルとして保存される請求項２に記載の分散ソフトウェアシステム。

【請求項４】　前記実行時システムは、さらに、前記第１のベースから前記第２のベースへのタスクの移動を容易にする請求項２に記載の分散ソフトウェアシステム。

【請求項５】　前記第１のオブジェクトはタスクであり、前記第２のオブジェクトはデータオブジェクトである請求項１に記載の分散ソフトウェアシステム。

【請求項６】　前記１以上のエージェントは、さらに、グローバルオブジェクト空間を有し、

該グローバルオブジェクト空間は、前記１以上のエージェント内のオブジェクトの記号参照符の、前記オブジェクトの対応する物理アドレスへのマッピングを有し、

それによって、前記第１のベースにある前記第１のオブジェクトは、前記第１のオブジェクトからの前記第２のオブジェクトへの記号参照符を取得し、前記オブジェクト空間の前記マッピングを用いて前記第２のオブジェクトの前記対応する物理アドレスを取得することにより、前記第２のベースにある前記第２のオブジェクトの前記物理アドレスについて知らなくとも、前記第２のベースにある前記第２のオブジェクトにアクセスを行う請求項１に記載の分散ソフトウェアシステム。

【請求項７】　前記オブジェクト空間は、各オブジェクトのアドレス指定用のグローバル識別記号を用いて実装される請求項６に記載の分散ソフトウェアシステム。

【請求項８】　前記第１のオブジェクトによる前記第２のオブジェクトの前記アクセスは、１以上の引数および一つの返送値を指定するメソッド呼び出しであり該１以上の引数または返送値への記号参照符を呼び出されたメソッドに送るかまたは該メソッドから送り返すことで、該１以上の引数または返送値を識別し、該１以上の引数または返送値を識別して、前記１以上の引数または返送

値の前記物理アドレスを呼び出されたメソッドに送るかまたは該メソッドから送り返す必要をなくすことで、メソッド呼び出しをネットワーク透明にする請求項１に記載の分散ソフトウェアシステム。

【請求項９】　前記１以上のエージェントは、さらにグローバルオブジェクト空間を有し、

該グローバルオブジェクト空間は、前記１以上のエージェント内のオブジェクトの記号参照符の、前記オブジェクトの対応する物理アドレスへのマッピングを有し、それによって分散ソフトウェアシステムは、前記呼び出されたメソッドに送ったりまたは該メソッドから送り返す前記１以上の引数または返送値に対する前記記号参照符を、

前記第１のオブジェクトから前記第２のオブジェクトへの記号参照符を取得し、

前記オブジェクト空間の前記マッピングを用いて前記第２のオブジェクトの前記対応する物理アドレスを取得することにより、識別することができる請求項８に記載の分散ソフトウェアシステム。

【請求項１０】　前記オブジェクト空間は、各オブジェクトのアドレス指定用のグローバル識別記号を用いて実行される請求項９に記載の分散ソフトウェアシステム。

【請求項１１】　前記第１のオブジェクトは、前記１以上のエージェントの前記保護領域内にあって、前記第１のベース内で実行する第１のタスクであり、

前記第２のオブジェクトは、前記１以上のエージェントの前記保護領域内にあって、前記第２のベース内で実行する第２のタスクであって、前記第１のタスクおよび前記第２のタスクが同時に、それぞれ前記第１のベースおよび前記第２のベースにて、前記１以上のエージェントの同じ前記保護領域内で実行する請求項１に記載の分散ソフトウェアシステム。

【請求項１２】　各ベースは、複数のエージェントに対して同時に、前記ローカルアドレス空間および前記コンピュータリソースを提供することができる請求項１に記載の分散ソフトウェアシステム。

【請求項１３】　前記第１のベースと前記第２のベースとは、異質コンピュータマシンにある請求項１に記載の分散ソフトウェアシステム。

【請求項１４】　保護領域を有してなる１以上のエージェントであって、該１以上のエージェントの前記保護領域が複数のコンピュータマシンのうちの２以上にあるエージェントと、分散ソフトウェアシステムのプログラマーによって前記２以上のコンピュータマシン間で選択的に移動可能である前記１以上のエージェントの前記保護領域内にある複数のオブジェクトであって、第１のオブジェクトが前記２以上のコンピュータマシン中の第１のコンピュータマシンにあり、第２のオブジェクトが前記２以上のコンピュータマシン中の第２のコンピュータマシンにあり、前記第１のコンピュータマシンにある前記第１のオブジェクトは、前記第２のコンピュータマシンにある前記第２のオブジェクトの物理アドレスがわからなくとも、そして前記第１および第２のコンピュータマシン間での前記第１のオブジェクト又は前記第２のオブジェクトのいずれの選択的移動とも無関係に、前記第２のコンピュータマシンにある前記第２のオブジェクトにアクセスすることができるオブジェクトとから構成される、ネットワークとして接続された複数のコンピュータマシンで用いるための分散ソフトウェアシステム。

【請求項１５】　前記第１のオブジェクトは、タスクであり、前記第２のオブジェクトは、データオブジェクトである請求項１４に記載の分散ソフトウェアシステム。

【請求項１６】　前記１以上のエージェントは、さらにグローバルオブジェクト空間を有し、

該グローバルオブジェクト空間は前記１以上のエージェント内のオブジェクトの記号参照符の、前記オブジェクトの対応する物理アドレスへのマッピングを有し、

それによって、前記第１のコンピュータマシンにある前記第１のオブジェクトは、前記第１のオブジェクトからの前記第２のオブジェクトへの記号参照符を取得し、前記オブジェクト空間の前記マッピングを用いて前記第２のオブジェクトの前記対応する物理アドレスを取得することにより、前記第２のコンピュータマシンにある前記第２のオブジェクトの前記物理アドレスについて知らなくとも、前記第２のコンピュータマシンにある前記第２のオブジェクトにアクセスを行う請求項１４に記載の分散ソフトウェアシステム。

【請求項１７】　前記オブジェクト空間は、各オブジェクトのアドレス指定用のグローバル識別記号を用いて実装される請求項１６に記載の分散ソフトウェアシステム。

【請求項１８】　第１のオブジェクトによる第２のオブジェクトのアクセスが引数および返送値のうちの１以上を指定するメソッド呼び出しであり、

該１以上の引数または返送値への記号参照符を呼び出されたメソッドに送るかまたは該方法から送り返すことで、該１以上の引数または返送値を識別し、

該１以上の引数または返送値を識別して、前記１以上の引数または返送値の前記物理アドレスを呼び出されたメソッドに送ったり該メソッドから送り返す必要を無くしてメソッド呼び出しをネットワーク透明にする、請求項１４に記載の分散ソフトウェアシステム。

【請求項１９】　前記１以上のエージェントは、さらに、グローバルオブジェクト空間を有し、

該グローバルオブジェクト空間は、前記１以上のエージェント内のオブジェクトの記号参照符の、前記オブジェクトの対応する物理アドレスへのマッピングを有し、

それによって、分散ソフトウェアシステムは、前記第１のオブジェクトからの前記第２のオブジェクトへの記号参照符を取得し、前記オブジェクト空間の前記マッピン

グを用いて前記第２のオブジェクトの前記対応する物理アドレスを取得することにより、前記メソッド呼び出しの前記１以上の引数または返送値に対する前記記号参照符を前記呼び出されたメソッドに送ったりまたは該メソッドから送り返すようにすることができる請求項１８に記載の分散ソフトウェアシステム。

【請求項２０】　前記オブジェクト空間は、各オブジェクトのアドレス指定を行うためのグローバル識別記号を用いて実行される請求項１９に記載の分散ソフトウェアシステム。

【請求項２１】　前記第１のオブジェクトは、前記１以上のエージェントの前記保護領域内にあって、前記第１のコンピュータマシン内で実行する第１のタスクであり、

前記第２のオブジェクトは、前記１以上のエージェントの前記保護領域内にあって、前記第２のコンピュータマシン内で実行する第２のタスクであって、前記第１のタスクおよび第２のタスクが同時に、それぞれ前記第１のコンピュータマシンおよび前記第２のコンピュータマシンにて、前記１以上のエージェントの同じ前記保護領域内で実行する請求項１に記載の分散ソフトウェアシステム。

【請求項２２】　さらに複数のベースを有し、

各ベースは、前記複数のコンピュータマシンのうちの１台にある１以上のエージェントに対してローカルアドレス空間およびコンピュータリソースを提供し、前記１以上のエージェントは、１以上のベースにある請求項１４に記載の分散ソフトウェアシステム。

【請求項２３】　前記１以上のエージェントは、少なくとも前記第１のコンピュータマシンの第１のベースにあって、加えて前記第２のコンピュータマシンの第２のベースにもあり、

前記第１のオブジェクトは、前記第１のベースにあり、

前記第２のオブジェクトは、前記第２のベースにある請求項２２に記載の分散ソフトウェアシステム。

【請求項２４】　各ベースが、複数のエージェントに対して同時に、前記ローカルアドレス空間および前記コンピュータリソースを提供することができる請求項２２に記載の分散ソフトウェアシステム。

【請求項２５】　各ベースは、オペレーティングシステムレベルのプロセスとして実行される請求項２２に記載の分散ソフトウェアシステム。

【請求項２６】　前記第１のコンピュータマシンと前記第２のコンピュータマシンとは、異質である請求項１４に記載の分散ソフトウェアシステム。

【請求項２７】　複数のベースがあり、前記ベースの各々が、複数のコンピュータマシンのうちの１台にローカルアドレス空間およびコンピュータリソースを提供し、１以上のエージェントがあり、前記エージェントの各々が保護領域を構成し、該１以上のエージェントの前記保護領域が複数のベースのうちの１以上のベースにあり前記複数のベースに関連する１以上の実行時システムがあり、前記複数のベース間でのエージェントおよびオブジェクトの移動を容易にする通信システムを有する実行時システムであり、前記ベース、エージェント、実行時システムとから構成される、ネットワークとして接続された複数のコンピュータマシンで用いるための分散ソフトウェアシステム。

【請求項２８】　各エージェントは、さらに、１以上のサブエージェントから構成され、

該各サブエージェントは、１つのベースにあり、

該サブエージェント中のオブジェクトを保存するオブジェクトメモリーと、

該サブエージェント中のタスクフレームを保存するタスクメモリーと、

該サブエージェントが属する前記エージェントのためのプログラムコードと、

サブエージェント制御ストレージとから構成され、

該サブエージェント制御ストレージは、

前記サブエージェントが属する前記エージェントを示すエージェントＩＤと、

オブジェクトの記号参照符を前記オブジェクトメモリー中で前記オブジェクトの対応する物理アドレスにマップするマッピングを有するオブジェクトテーブルと、

前記タスクメモリー中に複数のタスクスレッドポインタを保存するタスクスタックとから構成され、

前記１以上の実行時システムは、さらに、

対応するベースにあるサブエージェントの、複数のサブエージェント制御ストレージを管理する、各ベース用のエージェントマネージャと、

対応するベースにある複数のサブエージェントのための、複数のオブジェクトメモリーを管理する、各ベース用のオブジェクトマネージャと、

前記対応するベース以外の１以上のベースに、前記ネットワークによって前記オブジェクトを伝送するための、前記オブジェクトを直列化する各ベース用のオブジェクト直列化器と、

プログラムコードを読み、タスクメモリーにタスクスタックを作り、前記プログラムコードを実行する、各ベース用のタスクエグゼクタと、

前記対応するベース以外の１以上のベースに、前記ネットワークによって前記タスクスタックを伝送するための、前記タスクスタックを直列化する各ベース用のタスク直列化器と、

前記対応するベース以外の１以上のベースでタスクエグゼクタから遠隔オブジェクトアクセスメッセージを受け取り、前記対応するベース以外の１以上のベースに対して遠隔オブジェクトアクセス要求を送る、各ベース用の遠隔アクセスコントローラと、

前記コンピュータマシンと他の前記コンピュータマシン

との間の物理的通信を調整する通信システムとから構成される請求項２７に記載の分散ソフトウェアシステム。

【請求項２９】　前記プログラムコードは、前記サブエージェント内にクラスファイルとして保存される請求項２８に記載の分散ソフトウェアシステム。

【請求項３０】　前記実行時システムは、さらに、前記複数ベース間でのタスクの移動を容易にする請求項２８に記載の分散ソフトウェアシステム。

【請求項３１】　前記１以上のエージェントは、さらに、第１のベースにある第１のサブエージェントおよび第２のベースにある第２のサブエージェントを同時に有し、前記第１のサブエージェントは前記第１のベースに残り、前記第２のサブエージェントは前記第３のベースに移動することによって、前記１以上のエージェントが部分的に第３のベースに移動する請求項２７に記載の分散ソフトウェアシステム。

【請求項３２】　前記１以上のエージェントは、さらに、第１のベースにある第１のサブエージェントおよび第２のベースにある第２のサブエージェントを同時に有し、前記第１のサブエージェントおよび前記第２のサブエージェントの両方が併合して、一つのサブエージェントとして前記第３のベースに移動することで、前記１以上のエージェントが全体で第３のベースに移動する請求項２７に記載の分散ソフトウェアシステム。

【請求項３３】　第１のベースにある前記１以上のエージェントがあり前記分散ソフトウェアシステムは、さらに、１以上のアンカー固定オブジェクトを有し、
該１以上のアンカー固定オブジェクトは、ベース依存クラスから前記第１のベースで具体化されて、該第１のベースから他のベースには永久的に移動させることができず、
前記１以上のアンカー固定オブジェクトは、前記１以上のエージェントにあり前記１以上のエージェントには、第１の移動メソッドによって、前記第１のベースから第２のベースに移動するよう指示し、それによって、前記第１のアンカー固定オブジェクトは前記第１のベースに残り、前記１以上のエージェントのうちの残りは前記第２のベースに移動する請求項２７に記載の分散ソフトウェアシステム。

【請求項３４】　前記１以上のエージェントに、第２の移動メソッドによって、前記第１のベースから前記第２のベースに移動するよう指示し、それによって、前記第１のベースにある前記１以上のアンカー固定オブジェクトは、破棄され、前記１以上のエージェントのうちの残りは、前記第２のベースに移動する請求項３３に記載の分散ソフトウェアシステム。

【請求項３５】　前記１以上のエージェントは、第１のベースにあり、
前記分散ソフトウェアシステムは、さらに、１以上のアンカー固定オブジェクトを有し、
該１以上のアンカー固定オブジェクトは、ベース依存クラスから前記第１のベースで具体化されて、該第１のベースから他のベースには永久的に移動させることができず、
前記１以上のアンカー固定オブジェクトは、前記１以上のエージェントにあり前記１以上のエージェントには、第１の移動メソッドによって、前記第１のベースから第２のベースに移動するよう指示し、それによって、前記第１のアンカー固定オブジェクトは、破棄され、前記１以上のエージェントのうちの残りは、前記第２のベースに移動する請求項２７に記載の分散ソフトウェアシステム。

【請求項３６】　前記１以上のエージェントは、第１のベースにあり、
前記分散ソフトウェアシステムは、さらに、前記第１のベースから他のベースへの移動が一時的にできない１以上のピン固定オブジェクトを有し、
該１以上のピン固定オブジェクトは、前記１以上のエージェントにあり、
前記１以上のエージェントには、第１の移動メソッドによって、前記第１のベースから第２のベースに移動するよう指示し、それによって、前記第１のピン固定オブジェクトは、前記第１のベースに残り、前記１以上のエージェントのうちの残りは、前記第２のベースに移動する請求項２７に記載の分散ソフトウェアシステム。

【請求項３７】　前記１以上のエージェントには、第２の移動メソッドによって、前記第１のベースから前記第２のベースに移動するよう指示し、それによって、前記第１のベースにある前記第１のピン固定オブジェクトは破棄され、前記１以上のエージェントのうちの残りは、前記第２のベースに移動する請求項３６に記載の分散ソフトウェアシステム。

【請求項３８】　前記１以上のエージェントが第１のベースにあり、
前記分散ソフトウェアシステムは、さらに、前記第１のベースから他のベースへの移動が一時的にできない１以上のピン固定オブジェクトを有し、
該１以上のピン固定オブジェクトは、前記１以上のエージェントにあり、
前記１以上のエージェントには、第１の移動メソッドによって、前記第１のベースから第２のベースに移動するよう指示し、それによって、前記第１のベースにある前記第１のピン固定オブジェクトは破棄され、前記１以上のエージェントのうちの残りは前記第２のベースに移動する請求項２７に記載の分散ソフトウェアシステム。

【請求項３９】　前記１以上のピン固定オブジェクトをピン固定解除することで、該ピン固定解除オブジェクトを前記第１のベースから他のベースに移動できるようにして、それによって、前記１以上のエージェントが前記第１のベースから前記第２のベースに移動するよう指示

された時に、前記第１のベースにある前記ピン固定が解除されたオブジェクトが前記第２のベースに移動できるようにすることができる請求項３６に記載の分散ソフトウェアシステム。

【請求項４０】　第１のベースにある第１のエージェントが、第２のベースにある第２のエージェント中のオブジェクトに対する参照符を持ち、前記第１のエージェントが前記第１のエージェントの参照符と共に第３のベースに移動した後に、前記第１のエージェントの参照符が有効のまま残る請求項２７に記載の分散ソフトウェアシステム。

【請求項４１】　第１のベースにある第１のエージェントが、第２のベースにある第２のエージェント中の第１のオブジェクトに対する参照符を持ち、前記第２のエージェントが前記第１のオブジェクトと共に第３のベースに移動した後、第２のオブジェクトが作られて、前記第１のベースから前記第３のベースにある前記第１のオブジェクトの実際の位置への転送アクセスが可能となることで、前記第１のエージェントの参照符が有効のまま残る請求項２７に記載の分散ソフトウェアシステム。

【請求項４２】　第１のベースから、第２のベースにある第１のオブジェクトにメソッドを呼び出すための第１のメソッド呼び出しプロトコルであって、該メソッドが前記第１のベースから前記第２のベースに伝送され、該方法が前記第１のオブジェクトがある前記第２のベース上で実行されるプロトコルと、

前記第１のベースから、前記第２のベースにある第２のオブジェクトにメソッドを呼び出すための第２のメソッド呼び出しプロトコルであって、該メソッドが、前記第２のオブジェクトのメソッドに対応する前記第１のベース上のメソッドコードおよび前記第２のベース上の前記第２のオブジェクトに対する遠隔参照符を用いて、前記第１のベース上で実行されるプロトコルとを、

さらに有してなる請求項２７に記載の分散ソフトウェアシステム。

【請求項４３】　前記第２のオブジェクトの方法に対応する前記第１のベース上のメソッドコードは、前記第１のベース上のクラスファイルで保存される請求項４２に記載の分散ソフトウェアシステム。

【請求項４４】　オブジェクトクラスとエージェントクラスとベースクラスとタスククラスとを含む複数のオブジェクト指向クラスを定義する段階と、

選択されたオブジェクトインスタンスを前記ベースクラスで指定された位置まで移動させる前記オブジェクトクラスでのオブジェクト移動メソッドを定義する段階と、

エージェント内の全てのオブジェクトインスタンスおよびタスクインスタンスの移動を含む、選択されたエージェントプロセスを前記ベースクラスで指定された位置まで移動させる前記エージェントクラスでのエージェント移動メソッドを定義する段階と、

該エージェントクラスに従って第１のエージェントプロセスを具体化する段階であって、該第１のエージェントプロセスが複数のタスクインスタンスおよびオブジェクトインスタンスを含み、複数のコンピュータマシン間に分散している段階と、

前記第１のエージェントプロセス内で、前記オブジェクト移動メソッドと前記エージェント移動メソッドとを実行する段階とを有する、複数のコンピュータマシンからなるネットワークで用いるためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項４５】　さらに、タスクインスタンスに示される選択されたタスクを前記ベースクラスで指定された位置まで移動させる前記タスククラスでのタスク移動メソッドを定義する段階を有する請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項４６】　さらに、前記ベースクラスに従って複数のベースインスタンスを具体化する段階であって、前記第１のエージェントプロセスを２以上のベース上で同時に実行し、各ベースが前記複数のベースインスタンスのうちの一つによって指定される段階を有する請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項４７】　さらに、前記ベースクラスに従って複数のベースインスタンスを具体化する段階であって、前記第１のエージェントプロセスのオブジェクトインスタンスは２以上のベースに同時にあり、各ベースは前記複数のベースインスタンスのうちの一つによって指定される段階を有する請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項４８】　さらに、前記ベースクラスに従って複数のベースインスタンスを具体化する段階であって、前記第１のエージェントプロセスのタスクインスタンスは２以上のベースで同時に実行し、各ベースは前記複数のベースインスタンスのうちの一つによって指定される段階を有する請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項４９】　さらに、前記ベースクラスに従って第１のベースインスタンスを具体化する段階と、

前記エージェントクラスに従って第２のエージェントプロセスを具体化する段階であって、前記第１および第２のエージェントプロセスが、前記第１のベースインスタンスによって指定されたベース上で同時に実行する段階とを有する、

請求項４４に記載のコンピュータネットワークのための

ネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項５０】　さらに、第１のベースインスタンスによって指定された第１のベースにあるエージェントプロセスの選択された一部を、第２のベースインスタンスによって指定された第２のベースに移動させる前記エージェントクラスでの部分的エージェント移動メソッドであって、前記エージェントプロセスの前記選択された一部内の前記第１のベース上の全てのオブジェクトインスタンスおよびタスクインスタンスの移動を含むメソッドを定義する段階を有する請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項５１】　さらに、第１のベースインスタンスによって指定された第１のベースにある選択されたエージェントプロセス全体を、第２のベースインスタンスによって指定された第２のベースに移動させる前記エージェントクラスでの全体エージェント移動メソッドであって、前記選択されたエージェントプロセス全体内の前記第１のベース上の全てのオブジェクトインスタンスおよびタスクインスタンスの移動を含むメソッドを定義する段階を有する請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項５２】　さらに、前記ベースクラスに従って第１のベースを具体化する段階と、

前記エージェントクラスに従って第２のエージェントプロセスを具体化する段階であって、前記第２のエージェントプロセスが前記第１のベースに少なくとも部分的にある段階と、

ベース依存オブジェクトクラスからのアンカー固定オブジェクトを具体化する段階であって、該アンカー固定オブジェクトがベースインスタンスによって指定された第１のベースにある第２のエージェントプロセスと関連し、他のベースに移動させることができない段階と、

前記第２のエージェントプロセスをベースインスタンスによって指定された別のベースに移動させる前記エージェントクラスにおけるエージェント移動メソッドであって、前記アンカー固定オブジェクトを除く前記第２のエージェントプロセス内の全てのタスクインスタンスおよびオブジェクトインスタンスの移動を含む移動メソッドを定義する段階とを有する、

請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項５３】　さらに、前記ベースクラスに従って第１のベースを具体化する段階と、

前記エージェントクラスに従って第２のエージェントプロセスを具体化する段階であって、前記第２のエージェントプロセスが前記第１のベースに少なくとも部分的にある段階と、

前記オブジェクトクラスに従って第１のオブジェクトを具体化する段階であって、該第１のオブジェクトが前記第２のエージェントプロセス内にある段階と、

前記第１のオブジェクトを前記第１のベースにピン固定する段階と、

前記第２のエージェントプロセスを前記第１のベースから前記ベースクラスによって指定された別のベースに移動させる前記エージェントクラスにおけるエージェント移動メソッドであって、前記ピン固定された第１のオブジェクトを除く前記第２のエージェントプロセス内の全てのタスクインスタンスおよびオブジェクトインスタンスの移動を含む移動メソッドを定義する段階とを有する、

請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項５４】　さらに、前記ベースクラスに従って第１のベースおよび第２のベースを具体化する段階と、

前記第１のベース上で前記タスククラスに従ってタスクを具体化する段階と、

前記第２のベース上で前記オブジェクトクラスに従って、メソッドを有するオブジェクトを具体化する段階と、

前記第１のベース上のタスクからの前記第２のベース上の前記オブジェクトの前記メソッドの呼び出しが、前記第１のベースから前記第２のベースへの前記メソッドの伝送および前記第２のベース上での前記メソッドの実行を含む、メソッド呼び出しプロトコルを定義する段階とを有する、

請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項５５】　さらに、前記ベースクラスに従って第１のベースおよび第２のベースを具体化する段階と、

前記第１のベース上で前記タスククラスに従って、タスクを具体化する段階と、

前記第２のベース上で前記オブジェクトクラスに従って、メソッドを有するオブジェクトを具体化する段階と、

前記第１のベース上のタスクからの前記第２のベース上の前記オブジェクトの前記メソッドの呼び出しが、前記オブジェクトの前記メソッドに対応する前記第１のベース上のメソッドコードおよび前記第２のベース上の前記オブジェクトに対する遠隔参照符を使用しての前記第１のベース上での前記メソッドの実行を含む、メソッド呼び出しプロトコルを定義する段階とを有する、

請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項５６】　前記オブジェクトの前記メソッドに対応する前記第１のベース上の前記メソッドコードを、前記第１のベース上のクラスファイルに保存する請求項５５に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【請求項５７】　前記複数のコンピュータマシンの１以上のサブセットが異質である請求項４４に記載のコンピュータネットワークのためのネットワーク指向コンピュータソフトウェアのプログラミングシステムの実現手法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、分散および並列のコンピュータソフトウェアプログラミングの分野に関し、特に、プロセス移動性および通信について強化され、均質および異質のネットワーク環境のいずれにも適したネットワーク中心のアプリケーション構築に役立つ分散エージェントを含むソフトウェアシステムに関する。

【０００２】

【従来の技術】分散コンピュータシステムではこれまで、別個のコンピュータまたは「マシン」の集合間でのデータの分割および伝送に関する問題に主眼が置かれてきた。通常これらのシステムは、２つの方法のいずれかで、コードの分散およびアクセスを可能とするものである。例えば、クライアントサーバーシステムでは、各マシンは、そのマシンで見出されるリソースを制御するコードを保有している。他のシステムでは、同一のコード・イメージが全てのマシンに見られる。いずれの場合も、遠隔箇所での操作を呼び出すために、何らかの形のメッセージパッシングが用いられる。

【０００３】従来、プロセス移動性（すなわち、あるマシンから別のマシンへと実行プロセスを移動させること）は、あまり重要な問題ではなかった。クライアントサーバーに基づくシステムではプロセス移動性は本質的に無関係なものである。タスクは重く（すなわち、非常に多くの状態を有する）、制御リソースは特定のマシン上にある。全てのマシンが同一のコード・イメージを共有するシステムでは、プロセス移動性を用いて、局所性および負荷平衡を向上させることで、性能を高めることができる。しかしながら、タスクは通常重い処理を実行するために、タスクの移動は不可能なものとなっている。さらに、現在までに、単純で理解しやすい意味論を有する有効なタスク移動の方針は得られていない。

【０００４】最近、分散コンピュータシステムの実現において、プロセス移動性が徐々に重要になりつつある。プロセス移動性が向上することで、計算がそれ自体で再構成できるようになり、 データの局所性を高め、非局所通信事象が行われる回数を低減することができる。いくつかの分散システムモデルが開発されて、ある種のプロセス移動手段を提供している。

【０００５】

【発明が解決しようとする課題】命令形グルーシステム既存の命令プログラミング言語に対するシームレス拡張として動作し、その既存言語に対する分配および通信支援を加える命令形「グルー」システムが開発された。残念ながら、命令言語での計算には、まさに分散プログラムが回避すべき共有グローバルデータへの頻繁な変更を伴ってしまう。その問題を取り扱うべく、分散共有メモリー（すなわち、「ＤＳＭ」）および遠隔手続き呼び出し（すなわち、「ＲＰＣ」）という２つの基本的アプローチが開発された。

【０００６】ＤＳＭを用いた場合、計算の分散性はプログラマーにはほとんどわからないが、実装の複雑さは、メッセージパッシングを用いるシステムの場合より明らかに大きい。概念的には、あらゆるデータがグローバルアドレスに関連している。そこで、スレッド（thread）がどのマシンで実行されようとプログラムの挙動には影響を与えない。そこで、グローバルアドレスの参照解決は、そのアドレスを「所有する」マシンに対する遠隔通信の中で行われる。ＤＳＭは分散環境での命令言語の並列言語を実現する機構を提供するが、プログラマーは、一貫性および整合性がどのように実現されるかを制御できない。特に、データおよびタスクの分散はインプリメンテーションによって暗黙に処理され、プログラムによって明示的には管理されないことから、プロセス移動性の問題は、ほとんど無関係となる。ＤＳＭはプログラミングを単純化するものであるが、分配および通信を陽的に制御する機構と併用すると、さらに有効になると考えられる。

【０００７】ＲＰＣは、プログラムを、それぞれがそれ自体のアドレス空間で動く別個の部分に分割する方法を提供するものである。ＤＳＭの場合とは異なりＲＰＣ通信は、プログラムが陽的であることから、プログラマーはコストに対して完全な制御を行う。しかしながら、ＲＰＣの意味論は、通常の手続き呼出しの場合とはかなり異なっている。詳細には、手続きＰが手続きＱに対してＲＰＣ呼び出しを行う場合、Ｑに対する引数は直列化され、計算を実行すべきマシンに送られる。アプリケーションプログラムにリンクされた手続きに対するスタブジェネレータが、表現変換およびメッセージ化の取り扱いを担当する。遠隔手続きに送られた引数は、コピーによって送られる。それゆえ、呼出し元と呼び出し先との間の通信には、共用構造体に対する副作用はもはや使うことができない。結果的に、命令プログラムを大幅に変更して、ＲＰＣを使用する分散環境で動作するようにしなければならない。結果的に、ＲＰＣ意味論を用いる分散エージェントシステムのプログラミングは、逐次型マシンでの逐次プログラミングよりかなり複雑かつ巧妙なも

のとなる。

【０００８】プロセス移動性、すなわち一連の制御（またはタスク）をそれが関連する状態とともに移動させる能力は特に難しいものである。それらの言語が命令的性質のものであるということは、プログラム中で認められるかなりのパーセントのデータがグローバルでなければならないことを意味している。ＲＰＣを用いない場合、プロセス間の通信は副作用を介して行わなければならず、割り振りおよびコピーを介しては行わない。そこで、移動性プロセスを有することの利点は大きく低下する。概念的には、プロセスは状態を持たないことから、これらの言語では非常に移動性が高いが、グローバル（共用）データを頻繁に参照しなければならないことから、プロセス移動が有用なのは、それらプロセスがアクセスするデータが、そのデータを要求するプロセスとともに移動する場合のみである。グローバルデータはいくつかのプロセス間で共有されている可能性があることを考慮すると、命令言語でのデータおよびコードの暗黙的結合は、それらの言語におけるプロセス移動性の有用性を大きく低下させるものである。

ネットワーク中心的なオブジェクトに基づく言語

最近、新しい計算パラダイムへの移行が起こっている。実行プログラムの位置を、物理的な単一プロセッサ上の単一のアドレス空間として、あるいは１組のプロセッサ間に分配された独立のプログラムの集合体として見なすのではなく、Ｊａｖａのような並列性を持ったネットワーク指向のオブジェクトに基づく言語が登場することで、強力な選択肢が提供されるようになった（J.Gosling et al.,The Java Language, Specification, Sun Microsystems, Inc.. (1995)参照；当該参考資料は明白に本明細書に含まれるものとする）。可搬性のある分散型バーチャルマシンの上で同時に複数の制御スレッドを実行できるようにすることで、Ｊａｖａのようなネットワークを意識した言語は、異質プロセッサ集合間で単一のプログラムをシームレスに分配することができるような計算手法を提示する。実行前に同一コードが全てのマシンに常駐していることが必要な分散システムとは異なり、Ｊａｖａのようなコード移動型言語によって、新たなコードを既存の実行プロセスに転送しリンクさせることができる。この特徴によって、従来の分散システムでは不可能な形で、動的なアップロード機能を行うことができる。Ｊａｖａは、「オブジェクト」として知られる計算ユニットを組み込んでいる。一つのオブジェクトには、インスタンス変数と呼ばれるデータ集合とそのインスタンス変数に基づいて動作するメソッドと呼ばれる一組の操作が含まれる。オブジェクトの状態（すなわち、インスタント変数）は、オブジェクトの外部から、公然と見ることができるメソッドによってアクセスおよび操作される。このオブジェクト指向プログラムは自然な形のカプセル化を提供することから、分散環境に非常に適している。オブジェクトは、共有のリソースおよびサービスに対して規制されたアクセスを提供する。分散グルー言語とは対照的に、Ｊａｖａの分散拡張によって、基本型と同様にオブジェクトも通信することができる。さらに、Ｊａｖａ／ＲＭＩなどのある種の実装によっては、コードを、遠隔位置にあるアドレス空間に動的にリンクさせることができる。

【０００９】Ｊａｖａの主要な目的は分散環境でのコード移動を支援することにあることから（コード移動では、移動するコード中の命令が操作するデータの移動については仮定していないことから、コード移動はプロセス移動性とは概念的に異なるものであることは留意すべき点である）、その言語は、分散ネットワークにおける異なるマシン上のプロセスが通信を行うことができるよう、ソケット機構を提供する。しかしながらソケットは、低レベルのネットワーク通信の抽象化である。ソケットを用いるアプリケーションは、そのネットワーク層の上にアプリケーションレベルのプロトコルを積み重ねなければならない。アプリケーションレベルのプロトコルは、メッセージのエンコードおよびデコード、型のチェックや照合の実行などを担当する。その処理手順は誤りを犯しやすく、扱いにくいことが認められている。さらに、Ｊａｖａは、全体プログラムの移動のみをサポートしている。異なるマシン間では、制御スレッドを転送することはできない。

【００１０】ＲＰＣは、ソケットを使用する上で必要な低レベルの詳細を抽象化する一つの方法を提供する。しかしながらＲＰＣは、オブジェクトオリエンテッドシステムに対する適合性が低い。例えばＪａｖａでは、通信はオブジェクト間で起こり、手続き自体では起こらない。例えばウォルラスらの報告（A.Wollrath et al., "Java-Centric Distributed Computing" IEEE Micro, Vol.2, No.72, pp.44-53(May 1997)）に記載されているＪａｖａ／ＲＭＩは、Ｊａｖａの逐次実行コアによって定義されるオブジェクト意味論用に作られたＲＰＣの一種である。ローカルと遠隔の計算を分離するための基礎として手続き呼び出しを使用する代わりに、Ｊａｖａ／ＲＭＩはオブジェクトを使用する。遠隔計算は、遠隔オブジェクトについての手続きを呼び出すことで開始される。クライアントは、自身のマシンにある代用オブジェクトを介して遠隔オブジェクトにアクセスする。そのオブジェクトは、コンパイラによって自動的に発生するものであり、引数などの整列を扱うコードにコンパイルされる。他のあらゆるＪａｖａオブジェクト同様、遠隔オブジェクトはファーストクラスであり、手続き呼び出し引数として送ることができるか、あるいは手続き呼び出しからの結果として返送することができる。Ｊａｖａ／ＲＭＩは、命令言語の分散拡張または分散グルー言語では利用できない多くの特徴をサポートする。その中で最も重要なものは、クライアントとサーバーへのおよびク

ライアントとサーバーからの振る舞いを伝送できるというものである。何らかの抽象化を定義する遠隔インターフェースＩについて考えてみる。サーバーは、このインターフェースを実装して、ある特定の振る舞いを提供することができる。クライアントが最初にこのオブジェクトを要求すると、その実装を定義するコードを得る。すなわち、クライアントとサーバーが処理方針について一致している限りにおいて、その処理方針を実装するのに使用される特定の機構は動的に変更することができる。クライアントは、タスクとしてパッケージ化することでサーバーに振る舞いを送り、それをサーバー側で直接実行することができる。そして、実行する手続きがサーバー上に見つからない場合、それはクライアントから入手する。そうして遠隔インターフェースが、分散したマシン集合間で状態とともに実行可能な内容を動的に送るための強力な手段を提供する。Ｊａｖａ／ＲＭＩによって、Ｊａｖａ集合にあるマシン間でデータおよびコードの通信を行うことができる。そのような拡張によって、Ｊａｖａプログラマーは、マシン間を移動する単一のモノリシックユニット（アプレット（applets）の形等で）としてだけでなく、マシン集合間で区画される分散エンティティとして、計算を見ることができる。アーキテクチャー非依存型バーチャルマシンを使用することで、一つのプロセスからの情報を、各プロセスが実行しているマシンやそれらのピースを連結する基礎となるネットワークのインフラストラクチャーについてあまりわからなくとも、別のプロセスに送ることができる。

【００１１】しかしながら、Ｊａｖａ／ＲＭＩは使うのが困難な場合がある。遠隔オブジェクトは、暗黙的にグローバルハンドルまたはＩＤと関連していることから、ノードを介してコピーされることは決してない。しかしながら、遠隔オブジェクト手続き呼び出しで遠隔オブジェクトではない引数は、ほとんどＲＰＣの場合と同様にコピーされる。結果的に、遠隔呼び出しは、構文的には同一に見えたとしても、ローカル呼び出しとは異なる意味論を有する。Ｊａｖａが非常に命令的であるということは、分散プログラムを注意深く作成して、共有データの望ましくないコピーのために予想外の挙動が行われるのを回避しなければならないということである。

【００１２】さらに、ＪａｖａとＪａｖａ／ＲＭＩのいずれによっても、オブジェクトは、複数の異質マシンに同時に広がることはできない。各オブジェクトは、ある時間にはただ一つのマシンのみにある。結果的に、単一オブジェクトのカプセル化内にある複数マシンでの真の同時実行は不可能である。さらに、代表的なＲＰＣシステムのように、ＲＭＩを用いるタスク間の通信はコピーを介してのものである。そうして、Ｊａｖａ／ＲＭＩプログラムの意味論は、構文的に類似しているＪａｖａプログラムとは全く異なったものとなる場合がある。

エージェント言語

ＲＰＣおよびジャバ以外に、ネットワーク内で計算およびデータが自由に移動できるようにするエージェント言語に関して、多くの提案が行われている。概念的には、エージェントは計算（すなわちタスク）および移動性の（すなわち、マシンの分散ネットワーク内を自由に移動することができる）関連データのカプセル化である。

【００１３】例えば、アグレッツ（Aglets；例えば、D. B. Lange et al., "Programming Mobile Agents in Java-With The Java Aglet API", IBM, URL= http:///www.trl.ibm.com/aglets/agletbook, (1998) に記載）は、Ｊａｖａ／ＲＭＩインターフェースとセキュリティマネージャを用いて移動可能で安全なエージェントシステムを達成するＪａｖａ移動型エージェントシステムである。しかしながら、アグレッツによっては、エージェントの状態は、複数の異質マシンに同時に広がることはできず、エージェントの移動には、エージェント全体が一つのマシンから別のマシンへと移動する必要がある。

【００１４】他の移動型エージェント言語を２つ挙げるとテレスクリプト（Telescript）およびオデッセイ（Odyssey）がある（J. White, "Mobile Agents White Paper", General Magic, URL = http://www.generalmagic.com/technology/techwhite-paper.html (1998); "Introduction to the Odyssey API," General Magic, URL =http://www.generalmagic.com./agents/odysseyIntro.pdf (1998) 参照）。テレスクリプトもオデッセイもそのいずれのエージェントも実行中に移動することができるが、そのようなエージェントの状態は、ある瞬間には単一のマシンにしか存在できない。従って、テレスクリプトおよびオデッセイというエージェントでは分散した状態になることはできない。エージェントが移動する場合には、それの完全な状態が一緒に移動することが必要である。この制限によって、機能性と効率が大幅に低下する。オデッセイの場合、ヒープの中で見出される状態のみが移動することができ、スタック、プログラムカウンタおよびレジスタの状態はいずれも失われる。テレスクリプトにも同様の制限がある。

【００１５】移動計算を支援する他のシステムには、Ａｇｅｎｔ Ｔｃｌ（Agent Tcl；D. Kotz et al., "AGENT TCL": Targeting the Needs of Mobile Computers," IEEEInternet Computing, Vol. 1, No. 4, pp. 58-67 (1997) 参照）およびＡＲＡ（H. Peine et al., "The Architecture of the ARA Platform for Mobile Agents," Proceedings of the First International Workshop on Mobile Agents (K. Rothermel et al., eds.), pp. 50-61 (1997) 参照）があり、これらの基本言語は元々Ｔｃｌであったが、最近Ｊａｖａのサポートを得られるようになったものである。オデッセイおよびアグレッツの場合同様、これらのシステムでも、エージェントが分散状態を有することは禁止であり、あるエージェントが別のマシンにあるデータに透明にアクセスできるような基盤は

提供されない。

【００１６】異質ネットワーク内でのコードおよびデータの移動を可能とする他のプログラミング言語を２つ挙げると、オブリーク（Obliq；L.Cardelli, "A Language with Distributed Scope," Proceedings of the 22nd ACM Symposium on Principles of Programming Languages, pp.286-298 (1995) 参照）およびカリ（Kali；H.Cejtin et al., "Higher-Order Distributed Objects," ACM Transactions onProgramming Languages and Systems, Vol.17, No.5, pp.704-739 (1995) 参照）がある。オブリークの逐次的意味論は委任に基づくオブジェクトシステムであるが、カリはリスプ（Lisp）の高次のレキシカルスコープ型言語であるスキーム（Scheme；W.Clinger et al., eds. "Revised Report on the Algorithmic Language Scheme," ACM Lisp Pointers, Vol.4, No.3, pp.1-55 (July 1991) 参照）の上に構築されたものである。しかしながら、これら２つのいずれのシステムも、エージェントの明瞭な概念を持たない。オブリークは透明な参照を支援するものであるが、それは、オブジェクトが移動できる条件を厳しく制限することで行われるものである。さらに、オブリークは、エージェントなどの分散アドレス空間の概念を提供するものではない。カリは、遠隔参照での全ての操作が明示的に行われることを要求するものである。オブリーク同様にカリは、オブジェクトに基づくカプセル化モデルを支援しない。これらの制限によって、オブリークおよびカリは、移動型アプリケーションを有する大型の分散システムには適合しなくなる。

【００１７】従って、プログラミングが容易であって、（１）プロセスおよびエージェントの状態を複数の異質であっても良いマシンで分散させることができる、エージェントなどのオブジェクトに基づくカプセル化モデルを提供し、（２）別のマシンにあるデータの透明なアクセスを可能とし、（３）別個のマシン間で、全体もしくは部分的に、容易かつ有効なプロセス移動を可能とする、分散型計算システムがなお必要とされている。

【００１８】従って、本発明の主たる目的は、エージェントがそれのタスクおよび状態を、ネットワーク内で複数の異質であってもよい物理的マシン間に分散させることができる分散エージェントシステムを提供することにある。

【００１９】本発明の別の目的は、別個の物理的マシンにあるオブジェクトを含むエージェント内のオブジェクトへの参照で、そのオブジェクトの物理的な位置もしくはアドレスを知る必要がない、ネットワーク透明である分散エージェントシステムを提供することにある。

【００２０】本発明のさらに別の目的は、エージェント内でのオブジェクト参照が、プログラマーおよびエージェントに対して透明であるシステムによって解決される分散エージェントシステムを提供することにある。

【００２１】本発明のさらに別の目的は、選択可能で、位置独立の方法実行を提供する分散エージェントシステムを提供することにある。

【００２２】本発明のさらに別の目的は、別個のマシン間での全体もしくは部分的な容易かつ有効な実行時プロセス移動を可能とする分散エージェントシステムを提供することにある。

【００２３】本発明のさらに別の目的は、プログラミングが容易である分散エージェントシステムを提供することにある。

【００２４】本発明の他の目的は、添付の図面を参照しながらの以下の説明を考慮することで、より容易に明らかになろう。

【００２５】

【課題を解決するための手段】概して、本発明によれば、ネットワークとして接続された複数のコンピュータマシンで用いるための分散ソフトウェアシステムが提供される。そのシステムは、複数のベースを有し、各ベースは複数のコンピュータマシンのうちの１台上でローカルアドレス空間およびコンピュータリソースを提供する。保護領域を有してなる１以上のエージェントが提供され、その１以上のエージェントの保護領域は複数のベースのうちの１以上のベースにある。１以上のエージェントの保護領域内には、複数のオブジェクトが含まれており、第１のオブジェクトは複数のベース中の第１のベースにあり、第２のオブジェクトは複数のベース中の第２のベースにある。第１のベースにある第１のオブジェクトは、第２のベースにある第２のオブジェクトの物理アドレスがわからなくとも、その第２のベースにある第２のオブジェクトにアクセスすることができる。最後に、１以上の実行時システムが第１のベースおよび第２のベースに接続されている。その実行時システムは、少なくとも第１のベースから少なくとも第２のベースまでのエージェントおよびオブジェクトの移動をサポートするものである。

【００２６】別の実施態様では、システムは保護領域を有する１以上のエージェントを有してなることができ、その場合、１以上のエージェントの保護領域は複数のコンピュータマシンのうちの２以上のコンピュータマシンにある。１以上のエージェントの保護領域内には複数のオブジェクトが含まれ、第１のオブジェクトは上記２以上のコンピュータマシンのうちの第１のマシンにあり、第２のオブジェクトは上記２以上のコンピュータマシンのうちの第２のマシンにある。オブジェクトは、システムのプログラマーによって、２以上のコンピュータマシン間で選択的に移動可能である。第１のコンピュータマシンにある第１のオブジェクトは、位置透明的またはネットワーク透明的な形で、第２のコンピュータマシンにある第２のオブジェクトにアクセスすることができる。すなわち、第２のコンピュータマシンにある第２のオブ

ジェクトの物理アドレスがわからなくとも、ならびに第１および第２のコンピュータマシン間で第１のオブジェクト又は第２のオブジェクトのいずれが選択的に移動するかとは無関係に、それを行うことができる。エージェントは移動可能であり、ネットワークにおける他のいかなるマシンにも、全体もしくは部分的に移動することができる。さらに、ネットワーク内のマシンは同質もしくは異質であることができる。

【００２７】本発明にはさらに、複数のコンピュータマシンを有するネットワーク用のネットワーク中心コンピュータソフトウェアプログラミングシステムを実現する方法が含まれる。その方法は、オブジェクトクラス、エージェントクラス、ベースクラスおよびタスククラスを含む複数のオブジェクト指向クラスを定義する段階；選択されたオブジェクトインスタンスをベースクラスで指定された位置まで移動させるオブジェクトクラスでのオブジェクト移動メソッドを定義する段階；タスクインスタンスに示される選択されたタスクをベースクラスで指定された位置まで移動させるタスククラスでのタスク移動メソッドを定義する段階；エージェント内の全てのオブジェクトインスタンスおよびタスクインスタンスの移動を含む、選択されたエージェントプロセスをベースクラスで指定された位置まで移動させるエージェントクラスでのエージェント移動メソッドを定義する段階；そのエージェントクラスに従って第１のエージェントプロセスを具体化する段階であって、第１のエージェントプロセスは複数のタスクインスタンスおよびオブジェクトインスタンスを有し、上記複数のコンピュータマシン間で分散している段階；ならびに、上記第１のエージェントプロセス内で、上記オブジェクト移動方法、上記タスク移動方法および上記エージェント移動方法を実行する段階を含むものである。そうして、本発明は、ネットワークの各種マシン間に分散したエージェントの一部もしくは全体の移動を提供するものである。従って、本発明の各分散エージェントは、ネットワークのマシン中の１台、数台もしくは多数台間に分散されて、操作の同時性を高くすることができ、しかも同時に、エージェント内のタスクおよびデータ（それ自体、ネットワークのマシン間に分散されていてもよい）を、そのネットワークで動作する他のタスクおよびデータならびにそのようなタスクおよびデータが存在する同一マシンでのそのタスクおよびデータによる障害から保護する、保護されたカプセル化ソフトウェア構造を維持することができる。そのようなエージェントの移動は、プロセス実行中であっても簡単であり、ネットワークを通じて一貫性を保っている。具体的には、エージェント自体に対する事前の通知なしに移動させた後に、他のエージェントがある特定のエージェントに対して継続してアクセスすることができる。

【００２８】【発明の実施の形態】一般に、本発明は、オブジェクト指向言語で実現される。例えば本発明は、Ｊａｖａ言語との関連で実行することができる。本明細書での説明を容易にするため、言語シンタックスおよびコア意味論の説明は、Ｊａｖａに似せた形式で行い、Ｊａｖａの用語を用いるが（例えば、「静的メソッド」、「インスタンスフィールド」、「Ｓｅｌｆ」など）、それらは文脈から明らかになろう。しかしながら、本発明は、Ｊａｖａでの実現に限定されるものではなく、いかなるオブジェクト指向言語であっても好適なものでは実現可能であることは留意すべき点である。

【００２９】以下の記載について理解しやすくする上で、次の定義が有用である。

【００３０】「オブジェクト」：オブジェクトとはクラスのインスタンスであり、あらゆる種類の情報についての基本的保存単位を代表するものである。

【００３１】「オブジェクト空間」：オブジェクト空間とは、互いに自由に参照することができるオブジェクトおよびタスクの集合である。

【００３２】「保護領域」：保護領域とは、その保護領域の外部で実行する計算によって、その中でのデータへの不正アクセスを防止するソフトウェア構造である。

基本的構造

先ず、本発明の特有の分散エージェントモデルの基本的な構成要素を描いた図１について説明する。マシン１０ａおよび１０ｂなどのような複数のノードまたはマシン１０が、通信インターフェース２０を介して互いに接続されていることで、ネットワーク２５を形成している。マシン１０ａおよび１０ｂは、同質もしくは異質のマシンであることができる。各マシンには、ベース３０ａ、３０ｂおよび３０ｃなどの１以上の「ベース」３０がある。エージェント４０ａ、４０ｂおよび４０ｃなどのエージェント４０もあり、それらの各エージェントは１以上のベース３０上で動作する。

【００３３】各ベース３０は固有の識別記号を有し、マシン１０にローカルアドレス空間およびリソースを提供する。各ベースは例えば、ＵＮＩＸプロセスまたはウィンドウズプロセスなどのオペレーティングシステムレベルのプロセスとして実現される。最も簡単な場合、ベースは一つのプロセッサ上で動作する。しかしながら、共有メモリーマイクロプロセッサで可能なように、実装対象のオペレーティングシステムによって複数プロセッサでプロセスを実行可能となる場合には、一つのベースを複数プロセッサ上で動作させることもできる。更に、複数のベースを同一のマシン上で動作させることもできる。例えば、図１に示すように、ベース３０ｂおよび３０ｃは同一のマシン１０ｂ上で動作している。

【００３４】各エージェント４０は、グローバルオブジェクト空間および保護領域の両方として作用する移動性ソフトウェア要素である。エージェント４０は、いずれ

も同一の一貫した固有のオブジェクト空間にある１群のオブジェクトを管理するものである。各エージェント４０は、単純なオブジェクト（データオブジェクトなど）ならびにスレッドまたは現在実行中のタスクの集合などのオブジェクト集合をカプセル化する。各エージェント４０が１以上のベース３０で動作したり、いくつかのエージェント４０もしくエージェントのいくつかの部分が同時に同一ベース３０に存在したりすることができる。従って、本発明のエージェントは、１以上のベースおよび１以上の異なるマシンにあることができる。従って、エージェントの状態自体が、異なるマシンの集合全体に分散することができる。そのようなエージェントを実行する上で必要な構成要素の詳細については、「実行時データ構造」のセクションで後述する。

【００３５】エージェントメタファーによってプログラマーは、自律的にタスクを実行する移動性コードシステムを書くことができる。先行技術のエージェントとは異なり、本発明のエージェントは、同時並行の分散したタスクおよびデータの両方をカプセル化する。すなわち、エージェントの状態を実際にネットワーク内に分散させることができる。一つのエージェント内でのデータへの参照では、データがある物理的位置が分かっている必要はないことから、エージェント内のオブジェクトに、ネットワーク透明的な形でアクセスすることができる。結果的に、ネットワーク指向ソフトウェアを書いて、維持することが容易になる。一つのエージェント内におけるオブジェクト間の参照は位置透明的またはネットワーク透明的であることから、本発明のエージェントは、分散ネットワーク環境で「共有メモリー」抽象化を提供するものと考えることができる。さらに、本発明のエージェントは、以下に詳細に考察するように、一つのエージェント中のタスクとデータへの他のエージェントからの透明アクセスをむしろ禁止するようなタスクとデータのカプセル化を行うことで、高度のモジュール性および保護機構をも提供するものである。

【００３６】さらに、ネットワーク内に複数のエージェントを作成することができ、ネットワークのマシン間でのエージェントおよびエージェントの一部分（「サブエージェント」と称する）の分散を、以下に詳細に考察するように、移動によって動的に変更することができる。さらに、各エージェントは複数のタスクを有することができることから、単一の移動性ソフトウェア要素（エージェント）が、異機種でもありうる別のマシンで同時にタスクを実行することができ、それによって、単一の移動性ソフトウェア要素内での同時並行性を高めることができる。

オブジェクト間の通信

同一エージェント内のオブジェクト間の通信に関しては（「エージェント内通信」）、本発明の各エージェントによって提供されるカプセル化および各エージェントの分散性が重要な利点を提供する。具体的には、特定エージェントのオブジェクト空間内のオブジェクトは、それがどのベースにあるかとは無関係に（従って、どのマシンにあるかとは無関係に）同一エージェント中の他のオブジェクトによって透明にアクセス可能である。すなわち、あるオブジェクトが、他のオブジェクトに対して、そのオブジェクトがネットワーク中の同一マシンにあるか何らかの他のマシンにあるかとは無関係に、そしてそのオブジェクトがある物理的位置について明瞭に分かっていなくとも、アクセスすることができる。

【００３７】異なるエージェントにあるオブジェクト間の通信（「エージェント間通信」）は、以下のように管理される。全てのオブジェクトにグローバル識別記号を割り当てる。しかしながら、特殊な「Ｒｅｍｏｔｅ」インターフェースを実装するオブジェクトのみが、それが所属するエージェントの外部からアクセスすることができる。Ｒｅｍｏｔｅインターフェースを実装するオブジェクトが遠隔手続きに対する引数として送られると、そのオブジェクトに対する遠隔参照符が渡される。他のオブジェクト（すなわち、Ｒｅｍｏｔｅインターフェースを実装しないオブジェクト）が引数として現れると、そのオブジェクトのコピーが送られる。そのようなＲｅｍｏｔｅインターフェースの意味論は、例えばＪａｖａ／ＲＭＩ仕様に類似したものであることができる。

【００３８】上記のエージェント内およびエージェント間通信配置を図２に示してある。その図では、２個のベース３０ｂおよび３０ｃがマシン１０ｂにあり、エージェント４０ａは異なるマシン１０ａおよび１０ｂでそれぞれ動作する複数のベース３０ａおよび３０ｂにある。各エージェント４０には、１以上のオブジェクト５０がある。Ｒｅｍｏｔｅインターフェースを実装するオブジェクトは、斜線のある矩形で表してあって記号５２を割り当ててあり、Ｒｅｍｏｔｅインターフェースを実装しないオブジェクトは、黒塗り矩形によって表してあって記号５４を割り当ててある。オブジェクトに対する遠隔参照符は、白抜きの矩形によって表してあって記号５６が割り当ててある。矢印は、参照符と参照符のオブジェクトの間の依存性を表すものである。

【００３９】エージェント内であるがベース間の通信は、図２の矢印ＡおよびＢによって示したような遠隔参照符によって行われ、あらゆる種類のオブジェクトに一貫性を持ってアクセスおよび修正を行うことができる。そうして遠隔参照符は、単にそれが参照する実際のオブジェクトに対するスタブとして見ることができる。この実装によって、遠隔参照符のアクセスは、そのオブジェクトが実際に存在するマシンへの通信を必然的に伴い、この通信は、関連データへのアクセスまたは新たな計算の開始を伝える。上述のように、別個のベースで認められる単一のエージェントの部分はサブエージェントとして知られるものである。サブエージェント内（すなわ

ち、同一ベースにあるエージェント内）では、同一サブエージェントで認められる共有データにローカル参照によってアクセスすることができる。

【００４０】他方、エージェント間の通信は、オブジェクトがＲｅｍｏｔｅインターフェースを実装する場合は遠隔参照符を用いて、もしくは他の形態の場合はコピーすることで、行うことができる。矢印Ｃは、別のエージェントにおけるＲｅｍｏｔｅインターフェースを有するオブジェクト５２ｃを参照していることから、この参照は有効であるが、矢印Ｄは別のエージェントにおけるＲｅｍｏｔｅインターフェースを持たないオブジェクト５４ｂを参照していることから、その参照は無効になるものと考えられる。その無効が生じるのを回避するためには、Ｒｅｍｏｔｅインターフェースを持たないオブジェクトが引数または手続き呼び出しの返送値として現れる場合には、必ずオブジェクトのコピーを送らなければならない。エージェント間の通信については、以下に詳細に説明する。

オブジェクト空間

概念的には、タスクは、システム中のマシンでのオブジェクトに対する遠隔参照符（例えば、オブジェクトの名称）とそのオブジェクトの物理的位置との間のマッピングを定義するグローバルオブジェクト空間によって、エージェント内のデータにアクセスする。

【００４１】図３には、オブジェクト空間の動作の概念的線図を示してある。単一のエージェントであるエージェントが、マシン１およびマシン２という２つのマシン上のベース１およびベース２という２つのベースに広がっている。オブジェクト５５がベース１およびマシン１のエージェントで作られると、それの識別名称および位置が、オブジェクト５５に対する記号（または遠隔）参照符５５’とともに、エージェントのオブジェクト空間７０に記録される。オブジェクト５５に対するその後の参照は、記号参照符５５’を用いて行うことができ、システムはそれを、オブジェクト空間７０に問い合わせ、オブジェクト５５についての該当する物理的位置に対する記号参照符５５’を解くことで処理する。そのように、オブジェクト空間７０は、複数の物理的マシンに広がるエージェントにおける要素に対する透明アクセスを可能とするものである。特に、マシン２のベース２におけるマシン１のオブジェクト５５への参照は、記号参照符５５’を用いて行うことができる。そうして、データへの参照（それがベースに対してローカルであるか、あるいは別のベース上に認められるかとは無関係に）では、そのデータが存在する物理的位置について分かっている必要はない。

【００４２】実際には、グローバルにアクセス可能な物理的オブジェクト空間（共有メモリーなど）を使用してエージェント内のデータへの全ての参照を仲介することは可能であるが、かなり高価なものになり得る。本発明で使用されるオブジェクト空間のより有効な表示は、オブジェクト空間についての一次アドレス指定法としてのグローバル識別記号を使用するものであり、それについては後述する（「グローバルＩＤ」のタイトルの小セクション参照）。

エージェントおよびオブジェクトの移動

本発明の特に有用な特徴はプログラムの移動性である。本発明の分散エージェントシステムは、エージェントおよびオブジェクトについてのいくつかのユーザーレベルの移動メソッド、ならびにスレッド用の１つのシステムレベルの移動メソッドを組み込んだものである。移動は、移動性を実現する重要な手段であることから、本発明には重要である。他のエージェントシステムとは異なり、分散エージェントメタファーの結果としての本発明によって、実行時に、単にエージェントではなく任意のオブジェクトをネットワーク上で自由に移動させることができる。このように、エージェントレベルとオブジェクトレベルの両方で、移動性が実現される。タスクおよびデータは、自身のエージェント形成に関連したネットワーク中のマシン間で、自由かつ動的に移動することができる。オブジェクトおよびエージェントを移動可能とすることで、本発明は、先行技術によってはこれまで達成されなかったような適応性および柔軟性を提供する。

【００４３】エージェント移動によって、本発明のエージェント全体が、単一の分割不能なステップで移動するようになる。エージェントが異なるベースで実行する複数のスレッドからなる時にエージェント移動メソッドが呼び出された場合は、対象となるベースに移動する前に、それら全てのスレッドが一つのベースに集められる。オブジェクト移動によって、内部データおよび関連するスレッドの移動が可能となる（詳細については、「実行時システム」という名称のセクションで後述する）。

【００４４】ある一定の状況では、望ましくない種類の移動もあることは留意すべき点である。そのような状況に対処するため、本発明は「アンカー固定」性を提供し、これを実装するクラスのインスタンスが、Ｉ／Ｏポートまたは既存ソフトウェアに対するインターフェースオブジェクトなど、プロセス固有のオブジェクトに静的に依存している場合に用いるようにする。これらのオブジェクトは、自身をカプセル化しているエージェントが移動した場合でも移動してはならない。本発明はさらにオブジェクトについての「ピン固定」性をも提供する。これはアンカー固定性に類似しているが、動的制約を表現するものである。例えば、あるオブジェクトが特定場所への重要な通信を一時的に必要とする場合、該オブジェクトは最初にその位置に移動し、ピン固定性を設定し、次に効率の良い通信を行う。その間、オブジェクトは移動することができない。オブジェクトが再度移動しなければならない場合には、最初にそれのピン固定性を

リセットしなければならない。

【００４５】上記のように、エージェント移動によって、アンカー固定またはピン固定オブジェクトを除いて、オブジェクトおよびスレッドなどのエージェントの要素の全てが移動するようになる。本発明は、弱い移動と完全移動という２種類のエージェント移動を提供するものである。エージェント移動は、図４（ａ）～４（ｃ）に示してある。図４（ａ）には、目標となるベース３０ｋに移動する前のベース３０ｊにあるエージェント４０ｊを示してある。エージェント４０ｊには、いくつかのオブジェクト５０があり、それには移動可能なオブジェクト５７およびアンカー固定オブジェクト５８がある。

【００４６】図４（ｂ）には、「弱い移動」で移動したエージェント４０ｊの結果を示してある。弱モードでは、アンカー固定またはピン固定オブジェクトは元の位置に留まり、遠隔参照符でアクセスされることで、一貫した値を維持する。この結果、図４（ｂ）に示したように、移動後はエージェント４０ｊはベース３０ｊとベース３０ｋの両方に広がる（各ベースでのエージェント４０ｊの部分はサブエージェントである）。移動可能なオブジェクト５７は、標的ベース３０ｋに移動するが、アンカー固定オブジェクト５８はベース３０ｊに留まり、ベース３０ｊにあるアンカー固定オブジェクトにアクセスするためのベース３０ｋにおけるエージェント４０ｊ内に遠隔参照符５９が作られる。エージェントが移動して出たベースに残っているアンカー固定またはピン固定オブジェクトについては、エージェントがファイアウォールを通って移動することから、ファイアウォールを超えてアクセスすることはできないと考えられるが、エージェントが元のベースに戻ると、それらのアンカー固定もしくはピン固定オブジェクトはエージェントに再接続することができる。

【００４７】図４（ｃ）には、「完全移動」で移動したエージェント４０ｊの結果を示してある。完全移動は、元のベースにある全てのアンカー固定またはピン固定オブジェクトを破棄するものである。従って、図４（ｃ）に示したように、エージェント４０ｊがベース３０ｊからベース３０ｋまで移動しており、それに伴って移動可能なオブジェクト５７も移動している。アンカー固定オブジェクト５８はベース３０ｊから破棄される。従って、破棄されたアンカー固定オブジェクト５８への遠隔参照符５９は、無効になると考えられる。完全移動は、エージェントがサービスプロバイダのベースのような、エージェントが移動時にオブジェクトを残しておくことを許容しないベースにある場合に有用である。完全移動は、全てのオブジェクトに対するネットワーク透明アクセスを維持するものではないことから、完全移動を用いる場合は若干の注意を払う必要がある。

【００４８】エージェント移動はエージェント全体を目標となるベースに移動させるが、「ベース固有のエージェント移動」は、そのエージェントのうちの指定されたベースにオブジェクトを移動させるだけである。これは、エージェントをいくつかのベース間に分散させ、プログラマーが特定のベースにあるオブジェクトを別のベースに移動させたい場合に有用である。もしこれらのオブジェクトが既に同じエージェントを含んでいるベースに移動する場合、そのベースに元々存在したオブジェクトとそのベースに新たに移動したオブジェクトは一つにまとめられる。

【００４９】オブジェクト移動は、オブジェクトが別のベースに移動するものであり、関連するオブジェクトおよびスレッドの移動も必要とし得るものである。例えば、移動するオブジェクトに関連するスレッドは、スレッドがそのオブジェクトでメソッドを実行する（または現在実行中である）場合に、オブジェクトが移動するベースまで暗黙のうちに移動する。スレッドまたは他のオブジェクトは、マシン間を自由に移動することができる。スレッドが現在実行中のエージェントのオブジェクト空間が移動の目標となるベースに既に広がっている場合、移動は、状態を目標となるベースのサブエージェント中にコピーするという単純なことである。しかしながら、目標となるベースにサブエージェントが存在しない場合、最初に新たなサブエージェントを作ってから移動を開始するようにしなければならない。

【００５０】ベースはサーバーによって管理されてもよい。サーバーは永久アドレスを有するサービスプロバイダーであり、実際にサービスを実行する他のベースへのサービス要求を処理するものである。例えば、図５に示したように、サーバー６０は、エージェント４０ｍをサーバー６０が管理するいくつかのベース３０のうちの一つに移動させるというサービス要求を扱うことができる。

【００５１】本発明のエージェントが提供する共有メモリー抽象化の結果として、タスクおよびデータはエージェント内で自由にマシン間を移動することができる。移動は意味論保存的である。スレッドまたはオブジェクトの移動は、処理能力の示唆を有するだけである。すなわち、全てのオブジェクトがグローバルな識別記号を有する。従って、本発明の意味論および実装は、先行技術より大きい表現統一性および機能性を提供するものである。上述のように、各分散エージェントのグローバルオブジェクト空間のために、オブジェクトの物理的位置またはアドレスが分かっている必要はない（すなわち、ソースプログラムで明示する必要がない）。従って、オブジェクトを移動させるためには、そのオブジェクトの移動後の存在場所を反映するように、オブジェクト空間を適切に更新する必要があるのみである。この同一の機構によって、本発明では、部分的移動（エージェント内のオブジェクトまたはタスクのサブセットなどのエージェ

ント全体より小さい部分の移動）が実行可能かつ非常に有効なものとなる。

【００５２】ネットワーク透明性は、オブジェクトの移動性を示唆するものである。本発明における移動性または移動可能オブジェクトは、プログラマーによってサブエージェント間を選択的に移動させることができる（サブエージェントは特定のベースにあるエージェントの一部である場合）。マシン間を移動する移動性オブジェクトでも、エージェントのオブジェクト空間によって管理されるそれのグローバル識別記号を用いてアクセスすることができる。分散共有メモリーシステムとは対照的に、マシン間でのタスクおよびデータの移動は、プログラマーによって明示的かつ選択的に制御することができ、エージェント内でのそのようなオブジェクトの透明アクセスは、エージェント内でマシン間でのオブジェクトの移動とは無関係に維持される。その意味において、本発明で提供されるエージェントモデルは、分散共有メモリーモデルよりかなり改善されたものである。

【００５３】さらに、エージェントの全体移動のみが可能であって、他のエージェントには移動したエージェントの居場所は分からず、ソースプログラムへの静的参照符の書き込みがなければそれと通信を行うことができなくなってしまう先行技術とは極めて対照的に、本発明によれば、システム内のいかなるエージェントであっても、他のエージェント（またはサブエージェント）のオブジェクト空間を単に調べるのみで、移動とは無関係にそのエージェントにアクセスすることができる。

【００５４】複数のコンピュータマシンを有するネットワーク間での本発明のネットワーク指向の移動の実現の例を図６に示してある。最初に段階Ｓ８０で、オブジェクトクラス、ベースクラス、エージェントクラスおよびタスククラスを含む複数のオブジェクト指向クラスが定義される。次に段階Ｓ８１で、オブジェクト移動メソッドがオブジェクトクラスで定義される。呼び出しがあると、そのオブジェクト移動メソッドは、選択されたオブジェクトインスタンスを、ベースクラス（すなわち、ベースインスタンス）によって指定された位置に移動させる。段階Ｓ８２では、タスク移動メソッドがタスククラスで定義され、呼び出しがあると、選択されたタスクインスタンスがベースクラスによって指定された位置に移動する。同様に段階Ｓ８３では、エージェント移動メソッドがエージェントクラスで定義され、呼び出しがあると、選択されたエージェントプロセスが、ベースクラスによって指定された位置に移動する。

【００５５】移動メソッドが定義された後、段階Ｓ８４で、エージェントプロセスが、エージェントクラスに従って具体化される。そのエージェントプロセスは、タスククラスに従って具体化されるタスクインスタンスならびにオブジェクトクラスに従って具体化されるオブジェクトインスタンスを含む、もしくはカプセル化することができる。段階Ｓ８４で具体化されたエージェントは、ネットワークの複数のコンピュータマシン間に分散されることから、タスクインスタンスおよびオブジェクトインスタンスもエージェントプロセス自体と同様に、ネットワークのコンピュータマシン間で分散することができる。段階Ｓ８５では、エージェントプロセス内でオブジェクト移動メソッド、タスク移動メソッドおよびエージェント移動メソッドが実行される。段階Ｓ８５で実行されるメソッドは特定の順序で行う必要はなく、それぞれの方法を所望に応じて複数回行うことが可能であることは留意すべき点である。さらに、所望に応じて、一部の移動方法のみを定義・実行することができる。

メソッド呼び出しモデル

オブジェクト指向言語では、オブジェクトは、データと、そのデータを操作するためのメソッドまたは手続きというものの集合を定義する。メソッド呼び出しは、いくつかの引数を伴ってメソッドを起動するものである。本発明で使用されるような分散オブジェクト指向言語では、メソッド呼び出しはマシンの境界にまたがることができる。すなわち、呼び出しが行われるマシンは、呼び出されるメソッドを含むオブジェクトがあるマシンとは異なっていることができる。

【００５６】本発明は、メソッド実行についての２つの異なる方途すなわちプロトコルを提供するものである。それらの２つのプロトコルは、メソッドの呼び出し元および呼び出し先オブジェクトが物理的に同一マシンにない場合があるという事実に由来するものである。その２つの呼び出しプロトコルについて説明する前に先ず、オブジェクトに対する高速アクセスおよび低速アクセスについて説明する。高速アクセスモードでは、オブジェクト空間でのオブジェクトの識別記号をチェックしたり参照解除することなく、オブジェクトフィールドにアクセスする。そのような操作は、そのオブジェクトが呼び出し元のベースに存在する場合にのみ有効である。その場合、オブジェクトへのアクセスは、通常のアドレス指定機構によって行われる。低速アクセスモードでは、オブジェクトのグローバル位置をオブジェクト空間を介してチェックし、それの構成要素の一つにアクセスする都度参照解決しなければならない。

【００５７】これら２つのアクセスモードを利用するため、本発明では以下の２つの呼び出しプロトコルを定義する。

【００５８】（１）「ＲＰＣモデル」（遠隔手続き呼び出しモデル）は高速アクセスモードを使用する。メソッドは、そのメソッドを所有するＳｅｌｆオブジェクトが存在するベースで動作することから、そのオブジェクトのフィールドアクセスは常に高速モードで行うことができる。オブジェクトのグローバル識別記号の参照解決は必要ない。

【００５９】（２）「呼び出し元モデル」は、低速アク

セスモードを実現するものである。メソッドは呼び出し元ベースで動作し、Ｓｅｌｆオブジェクトが同一ベースにあることを要求しないことから、フィールドアクセスは、実際にアクセスする前に参照解決を必要とする。呼び出し元モデルによって、コードを呼び出し位置または呼び出された位置で動作させることができる。すなわち、同一エージェント内の異なるマシンでそれを行うことができる。

【００６０】これらのプロトコルの一方または他方を使用することで効率に影響があるが、２つのプロトコルのいずれもプログラムの振る舞いには影響しない。

【００６１】ＲＰＣモデルおよび呼び出し元モデルを図７に示してある。単一のエージェントが２つのベースであるベース１およびベース２に広がっている。メソッドｍ０（）は、ベース２で動作している。メソッドｍ０（）は、ベース１のオブジェクトｘに関連するメソッドｘ．ｍ１（）を呼び出す。ＲＰＣモデル下では、メソッドｍ０（）の計算はベース２から、オブジェクトｘおよび関連するメソッドｘ．ｍ１（）があるベース１に移動する。すなわち、メソッドｘ．ｍ１（）は、それがベース２で動作するメソッドによって呼び出されたものであるにも関わらず、ベース１で実行される。命令文「this.v1=0」などのフィールドアクセスは、ベース間の通信を要求しないローカル計算として行うことができる。メソッドｘ．ｍ１（）が完了した後、制御はベース２でのメソッドｍ０（）に戻る。次に、ベース１上のオブジェクトｘと関連する第２のメソッドｘ．ｍ２（）が、呼び出し元モデル下に呼び出される。オブジェクトｘに対する遠隔参照符がベース２で作られ、メソッドｘ．ｍ２（）はベース１ではなくベース２で動作する。命令文「this.y=0」などのフィールドアクセスは、ベース１およびベース２間での通信の発生を要求する。

【００６２】メソッド呼び出しの特殊な場合に、コンストラクター（constructor）メソッドと、アンカー固定またはＲｅｍｏｔｅインターフェースのいずれかを実装するクラスのインスタンスメソッドがある。コンストラクターメソッドでは、位置依存の初期化が行われる可能性が考えられることから、そのメソッドは常にＲＰＣモデルで呼び出される。アンカー固定オブジェクトに対するインスタンスのネイティブメソッドは、インスタンスの意味がその位置に依存することから、常にＲＰＣモデルで呼び出さなければならない。遠隔オブジェクトに対するインターフェースメソッドは、実際のオブジェクトを有するエージェントが存在するベースの一つで呼び出される。

【００６３】呼び出し元メソッドは、デフォルトでは呼び出し元ベースで実行されるが、プログラマーはさらに、「@［ベース］」の表現を有するメソッド名を用いて、呼び出し元メソッド呼び出しを実行しなければならないベースを明示的に指定することができる。この呼び出しでは、呼び出しベースすなわちインスタンスがあるベースとメソッドを実行するベースとが異なるかもしれないが、インスタンスのメソッドおよびフィールドは常に低速アクセスモードでアクセスされることから、それによってはエラーは生じない。

【００６４】本発明の言語では、次のデフォルトの振る舞いを仮定することが望ましい。

【００６５】（１）別段の断りがない限り、メソッドは常に、呼び出し元モデルで呼び出される。

【００６６】（２）プログラマーがＲＰＣメソッド修飾子をある方法に対して指定すると、メソッドは高速モードで、Ｓｅｌｆオブジェクトのインスタンスフィールドに直接アクセスする。ただし、そのプロトコルの実現には、実行が関連するオブジェクトがあるベースに移動する必要がある。

【００６７】（３）ＲＰＣ修飾子はさらに、静的メソッドにも適用できる。その場合、静的メソッドがクラスオブジェクトのある位置で呼び出され、次にメソッドが高速モードで静的フィールドにアクセスする。

エージェント間の通信

本発明のエージェントは、例えばＪａｖａのＲＭＩとかなり類似した形で、遠隔オブジェクトのインターフェースメソッドを呼び出すことで、別のエージェントと通信を行う。第一段階として、遠隔エージェントから呼び出すことができるインスタンスメソッドのシグネチャを用いて、遠隔インターフェースを拡張したインターフェースを定義する。第二段階として、上記インターフェースを実装するクラスを、そのメソッドの実装とともに定義する。第三段階として、上記クラスから作られるインスタンスオブジェクトを、エージェント自体のレジストリまたはグローバルレジストリエージェントのいずれかに登録する。第四段階として、遠隔エージェントがレジストリ中のオブジェクトを調べ、実際のオブジェクトに対するオブジェクト参照符を受け取る。最後に遠隔エージェントが、常にエージェントにまたがる遠隔メソッド呼び出しに適用されるＲＰＣモデルで、遠隔オブジェクトのインスタンスメソッドを呼び出すことができる。

【００６８】オブジェクト参照符が遠隔エージェントに送られると、遠隔インスタンス呼び出しが、引数を介して、遠隔エージェントにより多くの遠隔オブジェクト参照符を送り、返送値で別の遠隔参照符を得ることができることから、別個のエージェントを多くのオブジェクト参照符で緊密に結びつけることができる。オブジェクトが遠隔インターフェースを実行する場合は、引数および返送値は参照符によって送られる。それ以外の場合には値によって送られる（オリジナルの深いコピー）。オブジェクトがコピーされる場合、そのオブジェクトにおけるフィールド値の一貫性はエージェント間で維持されない。

動的リンク

本発明によって、新たなクラス定義（すなわちコード）を、動作中のプログラムに動的に差し込むことができる。この機能によって、再実行の必要なく、アプリケーションの機能性を徐々に高めることができる。その機能を提供するクラスローダの構造は、新たなコード構造の動的リンクを提供する他の言語（例えば、スキームまたはＪａｖａ）に見られる関連する機構と類似している。しかしながら、分散オブジェクト空間の導入によって、従来の作業ではなかった問題が生じる。

【００６９】本発明の分散オブジェクト意味論のために、エージェントは、クラスのロード方法およびロード位置を制御する複数のクラスローダを有する。実行の最初に作られる第１のクラスローダは、エージェントが動作開始するベースにリンクされて、デフォルトによって、ユーザー定義のクラスがベースからロードされるようにする事が望まれる。しかしながら、あるオブジェクトがそのオブジェクトのクラスがまだロードされていない新たなベースに移動する場合、そのクラスはその新たなベースからはロードできず、ソースベース（オブジェクトが最初に作られたベース）から目標のベースまで転送されなければならない。

【００７０】クラスローダはさらに、クラスオブジェクトも管理する。全てのクラスオブジェクトがクラスローダと同位置のベースにある必要はないが、クラスローダは、クラスオブジェクトがすでに作られているか否か、あるいは実際のクラスオブジェクトがどこにあるかを把握して、各クラスがクラスローダ用に１つのみのクラスオブジェクトを有するという規則を遵守できるようにしなければならない。エージェントが動作開始するベースとは異なる特定のローカルディスクにユーザー定義クラスファイルがあり、プログラマーがそのクラスをロードしたい場合、プログラマーはベース依存のクラスローダを用いることができる。

【００７１】図８～１０には、クラスロードの３つの例を示してある。図８および９には相当するクラスファイルがロードされていないベースへのオブジェクト移動の２例を示してあり、図１０には新たなクラスオブジェクト作成を図示してある。

【００７２】図８には、コアライブラリクラス中のオブジェクトの移動を示してある。コアクラスライブラリは、プログラマーが変更できないシステムクラスを代表するものと考えることができる。コアクラスファイルは常に、ローカルディスクからロードすることができる。図８に示したように、ベース１はクラスローダとクラスオブジェクト（クラス１）およびそのクラスのインスタンスを保持している。このインスタンスがベース２に移動する場合、そのオブジェクトのメソッドに関するコードを有するクラスファイルをロードしなければならない。それを行うには、次の段階を行う。オブジェクトが移動した後（矢印Ａ）、ベース１に見られる定義クラスへの遠隔参照符がベース２にも作られる（矢印Ｂ）。同様に、クラスローダに対する遠隔参照符も作られる（矢印Ｃ）。クラス１はコアクラスであることから、それをローカルディスクから、ベース１に見られるマシンにロードして（矢印Ｄ）、クラス１のインスタンスにリンクさせることができる（矢印Ｅ）。

【００７３】図９には、ユーザー定義クラスでのオブジェクト移動を描いてある。この場合、ソースベースにクラスファイルがなければならないことから、クラスファイルを、そのクラスファイルが存在するディスクまたは移動のソースベースからロードしなければならない。図９には、後者の例が示してある。ベース１は、クラスローダとクラスオブジェクト（クラス２）およびそのクラスのインスタンスを保持している。クラス２インスタンスがベース２に移動する場合（矢印Ａ）、クラスオブジェクトに対する遠隔参照符（矢印Ｂ）およびクラスローダに対する遠隔参照符（矢印Ｃ）が確立される。クラスローダに対する遠隔参照符（矢印Ｃ）によって、ベース１で作られるクラスファイルのそれ以降の動的リンクをベース２に透明的にロードすることができる。クラスオブジェクトはグローバル状態の情報を保有している場合があることから、クラスオブジェクトに対する遠隔参照符（矢印Ｂ）が必要となる。次に、方法定義を含むクラスファイルをベース１からベース２にコピーする（矢印Ｄ）。次に、インスタンスオブジェクトをこのクラスファイルにリンクさせる（矢印Ｅ）。

【００７４】最後に、図１０には新たなクラスオブジェクトの作成を描いてある。この場合、ベース２での計算で、新たなクラスに対する参照符が作られる。ベース１は、クラスローダおよびクラスオブジェクトを保有する（クラス３）。ベース２のクラスローダは単に、ベース１のクラスローダに対する遠隔参照符（矢印Ａ）である。クラスローダはベース１が所有するファイルシステムから（矢印Ｂ）、ベース２のローカルファイルシステムにクラスファイルをロードする（矢印Ｃ）。次に、クラス３オブジェクトの新たなインスタンスがベース２に作られる。クラス３のクラス自体の新たなインスタンスも、ベース２に作られる。システム中の所定のクラスに対する固有の参照符がなければならない場合は、ベース２に作られたクラス３オブジェクトに対する遠隔参照符がベース１に確立される（矢印Ｄ）。従って、ベース１で開始されるクラス３へのそれ以降の参照符は、ベース２にあるクラス３オブジェクトで見られる静的フィールドおよび方法を参照するものである。

実行時システム

実行時システムは、ベースのデータ構造を管理し、本発明者らが本発明で記載の特殊機能を提供するものである。図１１に示したように、各ベースは、ある種の管理機能および通信支援を提供する、対応する実行時システムにつながっている。単一の通信システムを用いて、特

定のマシンにある全ての実行時システムを扱うことができる。エージェントには複数のサブエージェントがあっても良く、各サブエージェントは別個のベースにある。この場合、別個のベースにあるサブエージェントは、それらの個々のベースで見られる１個もしくは複数の実行時システムによって支援された通信システムと接続されている。

【００７５】図１２には、ベースおよびそれの実行時システムにおけるサブコンポーネントについて詳細に描いてある。ベースには、クラスファイル、オブジェクトメモリー、タスクメモリーおよびサブエージェント制御ストレージなどの複数のデータブロックがある。オブジェクトメモリーは、サブエージェント外部の遠隔オブジェクトに対して参照を行う参照オブジェクトなどを含む、サブエージェント中の全てのオブジェクトを記憶する。オブジェクトメモリーは、実行時システムでオブジェクトマネージャによって管理され、サブエージェント制御ストレージ中のオブジェクトテーブルによって指定される。タスクメモリーは、タスクエグゼクタ（executor）がタスク実行を管理するのに使用するスレッドフレームを記憶する。クラスファイルは、タスクエグゼクタがアクセスするプログラミングコードを保持する。サブエージェント制御ストレージはサブエージェントの管理情報を記憶する。サブエージェント制御ストレージにおけるエージェントＩＤは、サブエージェントが所属する特定エージェント（すなわち、サブエージェントが一部分を構成しているエージェント）を識別する。サブエージェント制御ストレージにおけるオブジェクトテーブルは、サブエージェント中のオブジェクトメモリーを指定するものである。サブエージェント制御ストレージにおけるタスクスタックは、タスクメモリーを指定して、サブエージェントの実行状態を維持する。

【００７６】エージェントマネージャは、サブエージェント制御ストレージを使用し、エージェントを具体化するタスクエグゼクタとの通信を行って、ベースにおけるサブエージェントを管理し、クラスファイル中のプログラムを実行し、オブジェクトメモリー中のオブジェクトを具体化し、タスクメモリー中の実行タスクスタックを管理する。タスクとオブジェクトのいずれも、エージェント内および異なるベースにあるサブエージェント間で自由に移動できることから、異なった種類でも良いマシン（すなわち、異質マシン）間でのオブジェクトおよびタスクの伝送を行うための何らかの機構が利用できなければならない。直列化は、内部ポインタを有する複雑なデータ構造（ツリーまたはグラフなど）をフラットオブジェクト（アレイなど）に転換するプロセスである。元々のポインタは、フラット化した表現でのインデックスに取り替えられ、受信エージェントでポインタとして再度具体化される。直列化器の実装は簡単であり、入力構造における循環構造が適切に認識されるようにするために、特殊な注意を払う必要があるのみである。

【００７７】タスクエグゼクタはさらに、エグゼクタがタスクオブジェクトを直列化する要求を行うタスク直列化器およびエグゼクタが遠隔方法を呼び出す要求を行う遠隔アクセスコントローラとも通信している。遠隔アクセスコントローラの一つの実装の詳細については後述する（「実行時データ構造」というタイトルのセクションで）。オブジェクトマネージャは、オブジェクトメモリー中のオブジェクトを管理し、特にはオブジェクトを具体化し、不要オブジェクトを再利用し、オブジェクト直列化の要求をオブジェクト直列化器に対して行うことで、上記のオブジェクト空間を実現する。通信システムは、ネットワークに接続されたマシンにおけるベース間の相互作用を仲介するものである。

【００７８】以上、エージェントおよびオブジェクトの移動問題について概要を説明したが（「エージェントおよびオブジェクト移動」というタイトルのセクション参照）、以下に追加の考察では、そのような移動での実行時システムの関与について特に説明する。

【００７９】図１３～１７にはエージェント移動の順番の例を示してある。図１３に示したように、１個のサブエージェントＡを有するエージェントは、図の左側のマシンＡにあるベースＡにあることができる。ベースＡタスクエグゼクタは、サブエージェントＡを有してなるエージェント上でエージェント移動メソッドの実行によって、サブエージェントＡをマシンＢのベースＢに移動させるよう指示される。ベースＡタスクエグゼクタは、ベースＡエージェントマネージャに対して、サブエージェントＡについてのエージェント制御データを取得して、それをマシンＡ通信システムに送るよう要求する。エージェント制御データは、移動するエージェントについてのヘッダ情報とそれのタスクおよびオブジェクトを含むものである。次に、ベースＡタスクエグゼクタがベースＡタスク直列化器に対して、タスクメモリーでサブエージェントＡ内のタスクオブジェクトを直列化するよう要求し、ベースＡタスク直列化器は直列化したタスクをマシンＡ通信システムに送る。同様に、オブジェクトも直列化され、ベースＡのオブジェクトマネージャおよびオブジェクト直列化器によって、マシンＡ通信システムに送られる。図１４に示したように、次に、マシンＡ通信システムは、ネットワークを通してサブエージェントＡについての直列化オブジェクト、直列化タスクおよびエージェント制御データを、マシンＢについての通信システムに送る。

【００８０】マシンＢ通信システムがサブエージェントＡについてのエージェント移動データ（エージェント制御データ、直列化タスクおよび直列化オブジェクトを含む）を受信した後、ベースＢエージェントマネージャがベースＢ上でサブエージェントＡについてのメモリーブロックを割り付け（サブエージェントＡ’と称する）、

サブエージェントＡ’について、ベースＢにサブエージェント制御ストレージを作る。マシンＢのタスクエグゼクタおよびオブジェクトマネージャも、それぞれベースＢのタスクメモリーおよびオブジェクトメモリー中にタスクオブジェクトとデータオブジェクトを作る。サブエージェントＡ’をそうしてベースＢで具体化した後に、図１５に示したように、ネットワークを通して、クラス要求がベースＢからベースＡまで送られる。図１６に示したように、ベースＡは、ベースＢでエージェントを再構築するのに必要なエージェントについてのクラスファイルをベースＢへネットワークによって送信することで、クラス要求に対応する。全ての移動段階が終了した後、図１７に示したように、ベースＡにおけるサブエージェントＡ用のメモリーブロックが解放され、エージェントはベースＢ上にサブエージェントＡ’として再構築される。この例では、マシンＡとマシンＢは異質であっても良い。タスクおよびオブジェクトの構造におけるマシンの依存性は、実行時システムによって、特にタスク直列化器およびオブジェクト直列化器によって解決される。

【００８１】図１８および１９には、ベースにあるエージェントの一部が別のベースに送られ、エージェントの残りの部分はそのまま現在のベースに残る部分エージェント移動の例を描いてある。この例では、図１８に示したように、エージェントはサブエージェントＡ１およびサブエージェントＡ２という２つのサブエージェントを有し、それらはそれぞれベースＡおよびベースＢという２つのベースにある。エージェントの部分移動が要求され、それによってサブエージェントＡ１のみが、ベースＡからベースＣに移動するよう要求されるが、サブエージェントＡ２はベースＢに残る。サブエージェントＡ１についての直列化プロセスは、図１３～１７に示し、前述のような方法と類似の方法で行われる。図１９に示したような部分移動の後、サブエージェントＡ１はサブエージェントＡ１’としてベースＡからベースＣに移動していることから、エージェント全体がベースＢおよびベースＣの両方にある。

【００８２】図２０および２１には、エージェントの全部分が目標のベースに移動する全エージェント移動の例を描いてある。この例では、図２０に示したように、エージェントにはサブエージェントＡ１およびサブエージェントＡ２という２つのサブエージェントがあり、それらはそれぞれベースＡおよびベースＢという２つのベースにある。ベースＡにあるサブエージェントＡ１は、サブエージェントＡが属するエージェント全体をベースＣに移動させるよう要求する全体エージェント移動方法を実行する。エージェントの別の部分、すなわちサブエージェントＡ２は偶発的に別のベース、すなわちベースＢにある。全体エージェント移動の結果、図２１に示したように、サブエージェントＡおよびサブエージェントＢのいずれもベースＣに移動し、単一のサブエージェントであるサブエージェントＡ３に併合される。

【００８３】図２２および２３には、オブジェクト移動の例を示してある。単一サブエージェントであるサブエージェントＡ１は、図２２に示したようにベースＡにある。サブエージェントＡ１には、オブジェクトＯ１という一つのオブジェクトを含むオブジェクトメモリーがある。プログラマーは、オブジェクトＯ１がベースＡからベースＢに移動するよう要求する。オブジェクトＯ１は、ベースＡの実行時システムおよび通信システムを用いて直列化され、ベースＢに送られる。ベースＢ通信システムが直列化オブジェクトＯ１を受信し、ベースＢにエージェントに関連するサブエージェントがない場合は、図２３に示したように、ベースＢ実行時システムは、新たなサブエージェントであるサブエージェントＡ２用の新たなメモリーブロックを作る。この例では、オブジェクトＯ１の移動後には、エージェントはベースＡおよびベースＢの両方にあり、転送オブジェクト「ｒ」がサブエージェントＡ１’で作られ、ベースＢのオブジェクトＯ１に送られて、オブジェクト移動後であっても、オブジェクトＯ１に対するネットワーク透明参照符を維持するようになっている。

【００８４】図２４および２５には、エージェント移動の場合の遠隔オブジェクトアクセスの１例を示してある。この例では、図２４に示したように、第１のエージェントであるエージェントＡがベースＡにあり、参照オブジェクトＲを有するベースＡにあり、その参照オブジェクトはオブジェクトＯ１を参照し、それはベースＢにある第２のエージェントであるエージェントＢ内にある。エージェントＢは第３のベースであるベースＣに移動し、それによってエージェントＢは、図２５に示したように、ベースＢにサブエージェントＢ１およびベースＣにサブエージェントＢ２を有することになる。転送オブジェクト「ｒ」がベースＢのサブエージェントＢ１で作られることで、図２５にも示したように、エージェントＢがベースＣに移動した後であっても、ベースＡはオブジェクトＯ１にアクセスすることができる。

【００８５】図２６および２７には、エージェント移動の場合の遠隔オブジェクトアクセスの別の例を示してある。この例では、図２６に示したように、第１のエージェントであるエージェントＡがベースＡにあり、オブジェクトＯ１を有するサブエージェントを含んでいる。第２のエージェントであるエージェントＢはベースＢにあり、ベースＡのオブジェクトＯ１を参照する参照オブジェクトＲを有するサブエージェントを含んでいる。エージェントＢは第３のベースであるベースＣに移動し、それによりエージェントＢは単一のサブエージェントを有することになり、それは図２７に示したように、この時点ではベースＣに存在する。ベースＣのエージェントＢ内に新たな参照オブジェクトＲ’が作られて、ベースＡ

にあるオブジェクト１への定常的アクセスが維持される。

実行時データ構造

本発明におけるインスタンスは、ヒープの中から割り付ける事が望ましい。

【００８６】６４ビット値が適切に整列されるのが望ましいため、全てのオブジェクトが、６４ビット境界を維持するようにする事が望ましい。バイトアドレス指定可能なマシンでは、これにより、３個までの下位ビットをタグとして用いることができる。整合性を得るため、可能な限り、実行時データ構造は本発明のインスタンスとして実装される。

【００８７】不要オブジェクト回収装置（garbage collector）およびメッセージを直列化させるコードのいずれも、ポインタとデータとを区別し、メモリ中のオブジェクトの大きさを決定する必要がある。直列化コードはさらに、追加情報を必要とする。例えば、インターンされた（interned：唯一であることが保証された）文字列とインターンされていない（uninterned： 唯一であることを保証されてない）文字列との間を区別できなければならない。異なるマシン間でデータを移動させる場合は、浮動小数点数を変換して、直列化コードがやはりそれらの位置決めをできるようにしなければならない場合がある。

【００８８】（ａ）インスタンス

各インスタンスは、それ自体のフィールド以外に、それのクラス、整数の「ハッシュコード」およびミューテックス（mutex）を有する事が望まれる。インスタンスが、それが作られたベースから外部に公開されたことがある場合には、グローバルＩＤも含まれる。ほとんどのＪａｖａの実装では、メモリーにおけるオブジェクトの位置から、インスタンスのハッシュ値を導き出す。本発明はベース間でオブジェクトを移動させ、これにともないメモリー中でのそれらの位置が変わることから、ハッシュ値をインスタンス自体に保存する必要がある。ハッシュコード、ミューテックスおよびグローバルＩＤは必要に応じて作られる。新たに作られたインスタンスは、それらのいずれも持たない。図２８の「インスタンス」ブロック２００に示したように、インスタンスのレイアウトは次のようにする事が望ましい。

【００８９】－インスタンスのクラス
－（整数のハッシュコード）
－（ミューテックス）
－（グローバルＩＤ）
－インスタンスのフィールド

ほとんどのオブジェクトが、ハッシュ、ロック、外部公開されていない。従って、これらオブジェクトの共通部分の大きさを小さくするため、それへのアクセスにおける若干のコスト上昇を犠牲にして、それらのフィールドを一つにまとめることができると考えられる。

【００９０】不要オブジェクト回収装置およびコード直列化器が必要とするデータレイアウト情報など、特定のクラスのインスタンスに共通する全ての情報は、そのクラスに保存される。

【００９１】（ｂ）アレイ

整合性を得るために、アレイは、特殊なアレイクラスのインスタンスとして表されることが望まれる。各インスタンスは、アレイの種類、大きさおよび構成要素を含むフィールドを有する。アレイのクラスによってアレイの大きさは決定されないことから、他のインスタンスとは異なりアレイインスタンスは、不要オブジェクト回収装置によって特別に処理する必要がある。

【００９２】（ｃ）クラス

図２８に示したように、クラスオブジェクト２１０は以下の５つの領域にまとめることができる。

【００９３】（１）不要オブジェクト回収装置（ＧＣ）用のクラス固有の情報；
（２）不要オブジェクト回収装置（ＧＣ）用のインスタンス固有の情報；
（３）全てのクラスに共通のデータ；
（４）インスタンスおよび静的メソッドのテーブル；
（５）定数および静的フィールド

全てのクラスに、以下のデータがある。

【００９４】－そのクラスについてのデータレイアウト情報
－それがアレイクラスであるか否かなどのそのクラスのインスタンスについてのデータレイアウト情報
－そのクラスのスーパクラス
－そのクラスについての「Ｃｌａｓｓ」クラスのインスタンス
－そのクラスをロードするのに使用されるクラスローダ
－初期化状態
－インターフェースメソッドテーブルインデックス

メソッドテーブルは、そのクラスにおける各インスタンスおよび静的メソッドについてのコードに対するポインタ列である。インスタンスは、適切なオフセットで認められるコードにジャンプすることで呼び出される。インスタンスメソッドコードのオフセットは、あるクラスとその任意のサブクラスについて同一でなければならないことから、いずれのクラスにおいても、インスタンスメソッドテーブルは同じオフセットで始まる。

【００９５】それが実装するクラスおよびインターフェースの名称は、クラスインスタンスにある。キャスト（cast）および実行時の型のチェックを高速化するため、各クラスには、クラス階層におけるそれの位置についての簡潔な表現を持たせることもできる。

【００９６】図２８には示していないが、クラスのメソッドについてのコードは、そのクラスに戻るポインタを有することができる。クラスオブジェクトおよびそれのコードはヒープには置かないことが望ましい。そうでは

なく、それらはクラスファイルの一部であり、クラスファイルをロードする時に作られる。

【００９７】（ｄ）スレッド

スレッドは、実行時におけるプログラムの実行状態を表現するものであり、スレッドクラスのインスタンスであることができる。そのクラスについての標準的フィールドに加えて、各スレッドはスタックを持つ。このスタックは、スタックセグメントのリンクされたリストであり、その各セグメントには、スタックフレーム列がある。フレームの実装には、ローカル変数および引数のための大きさと型情報へのポインタがある。その情報は型チェックのルーチンや不要オブジェクト回収装置に使用されることが望まれる。もし不要オブジェクト回収がまれにしか起こらない場合には、この情報を動的に生成することも出来る。

【００９８】（ｅ）サブエージェント

サブエージェントは、特定ベースにあるエージェントの一部である。サブエージェント内のインスタンスは、そのサブエージェントにとっては「ローカル」である。他の全てのインスタンスは「遠隔」である。サブエージェントは、サブエージェントクラスのインスタントとして表される。それのフィールドおよびメソッドはいずれも、通信プロトコルに関係するものであり、そのセクションに詳述してある。

【００９９】（ｆ）遠隔参照符

他のサブエージェントに存在するインスタンスに対する参照は、ローカルインスタンスとほとんど同じ表現を有する。クラスポインタは通常のクラスを指定するのではなく、メソッドポインタが本来のメソッドのＲＰＣスタブを指定するような本来のクラスのコピーを指定する。遠隔インスタンスについてのメソッドの呼び出しは、ローカルインスタンスにおけるメソッドの呼び出しと同じである。それは、メソッドディスパッチを行う場合に、オブジェクトの位置を調べる必要性を回避するものである。そのようなチェックは、Ｓｅｌｆ以外のインスタンス用フィールド参照を行う時に必要である。遠隔参照符にはフィールドはなく、空白ではないグローバルＩＤを有する。

【０１００】（ｇ）グローバルＩＤ

グローバル識別記号すなわち「グローバルＩＤ」は、他の１以上のベースに参照されたインスタンスの、名前と現在の位置を記録する。そのインスタンスのグローバルな名称は、それが作られたベースと、そのベースによって割り当てられた整数識別記号によって決定される。

【０１０１】グローバル識別記号は、オブジェクト空間を実装するための機構である。エージェント内のいずれのオブジェクトにもグローバルＩＤがある。その識別記号の内容は、それがシステムのどこにあるかとは無関係に、オブジェクトの位置を決定できるものである。

【０１０２】グローバルＩＤは、以下のデータを有する事が望まれる。

【０１０３】－整数の識別記号

－インスタンスが属するサブエージェント

－「空白」またはそれが現在ローカルベースにある場合にはインスタンス

－参照カウントおよびグローバル不要オブジェクト回収装置によって必要とされる他の情報

オブジェクトがそれの元の場所から移動する場合には、転送ポインタが必要である。転送ポインタを指す参照符は、オブジェクトの新たな位置を反映するように更新される。

【０１０４】（ｈ）通信プロトコル

本発明に適した通信プロトコルの実装の１例について以下に考察する。しかしながら、本発明との関連で使用が好適である他の好適な通信プロトコルを作り上げることが可能であること、ならびに本発明が以下に記載の特定の通信プロトコルによって制限されないことは留意すべき点である。

【０１０５】このプロトコルは元々、カリ（Kali）言語（H. Cejtin et al., "Higher-OrderDistributed Objects," ACM Transactions on Programming Languages and Systems Vol. 17, No. 5, pp. 704-739 (1995) に記載）用に設計・実行されたものであり、１９９８年４月２８日発行の米国特許５７４５７０３号（発明の名称「Transmission Of Higher-Order Objects Across A Network Of Heterogeneous Machines」）に記載されている。これらいずれの引例も、記載することで本明細書に組み込まれるものとする。カリのインプリメンテーションの多くを用いて、本発明用の通信プロトコルを実行することができる。

【０１０６】（１）<U>共有データ構造

ほとんどのインスタンスが一つのサブエージェントのみに存在する。他のいずれのサブエージェントにおいても、そのインスタンスはフィールドその他のデータを持たない遠隔参照符によって表されている。その規則にはいくつかの例外があり、それはクラス、インターンされた文字列およびサブエージェントである。

【０１０７】いずれのサブエージェントも、あらゆるクラスの静的データのローカルコピーを有する。クラスの非定数の静的フィールドの値は、一つのサブエージェント上にある。

【０１０８】全ての文字列および文字列値の定数式はグローバル識別記号を有する。各サブエージェントは、それが参照する全てのインターンされた文字列についてのそれ自体のコピーを有する。文字列には、変化するデータはないことから、混乱は生じない。

【０１０９】他のサブエージェントのローカル表現には、そのサブエージェントと通信を行う上で必要な情報が含まれていなければならない。クラスおよびインターンされた文字列とは異なり、そのデータはローカルであ

り、他のサブエージェントにある情報のコピーではない。サブエージェントの構造について、次のセクションで説明する。

【０１１０】（２）サブエージェントデータ

全てのサブエージェントインスタンスもグローバルＩＤを有する。全てのサブエージェントインスタンスが以下のフィールドを有する事が望ましい。

【０１１１】－グローバルに固有の識別記号

－解析ベクトル：インスタンスに対するＩＤをマッピングするベクトル

－保留ベクトル：部分的に変換されたインスタンスに対するＩＤのマッピングのベクトル

特定のサブエージェントが通信を行っているサブエージェントインスタンス中のフィールドは下記の通り。

【０１１２】－ベース：サブエージェントがあるベース

－待ち行列：接続を確立するために待つスレッド列

－入力ポート、出力ポート：サブエージェントと対話するためのポート

（３）通信インスタンス

インスタンスは、そのインスタンスのクラスのＩＤ、そのインスタンスを作ったサブエージェントのＩＤおよびインスタンス自体のＩＤという３つのＩＤとして転送されることが望ましい。インスタンスがローカルサブエージェントによって作られ、まだ転送されたことがない場合、それに対するグローバルＩＤを作って、インスタンスをローカルサブエージェントの解読ベクトルに加えなければならない。

【０１１３】サブエージェントインスタンスの全てのＩＤが、そのサブエージェントを代表するローカルインスタンスのＩＤであることに留意する。特定のサブエージェントには、それを知っている全ての他のサブエージェントによって異なるＩＤが割り当てられる場合がある。変換によって、ローカルサブエージェント自体のＩＤはゼロとする事が望ましい。

【０１１４】受信側サブエージェントは、上記の３つのＩＤを次のように用いる。サブエージェントＩＤを転送側エージェントについての解析ベクトル中で調べ、インスタンスＩＤをそのサブエージェントの解析ベクトルで調べる。クラスＩＤは、第２の検索が不首尾である場合にのみ使用する。

【０１１５】例えば、サブエージェントＡ、ＢおよびＣについて考え、ＡがＢに対してＩＤ「３」を割り当てているものと考える。さらに、Ｂはインスタンスｉを有し、そのインスタンスに対してはＩＤ「２」を割り当てており、それをＡに送信している。サブエージェントＡがサブエージェントＣに対してｉについての参照を送る場合、サブエージェントＡはＩＤ「３」（サブエージェントについてのＩＤ）およびＩＤ「２」（インスタンスについてのＩＤ）を送る。次に、サブエージェントＣがサブエージェントＡに関するそれの解読ベクトルを用いて、「２」をＢに関するそれのサブエージェントインスタンスに翻訳し、次にそのインスタンス中の解読ベクトルを用いて、「３」をローカル参照符に翻訳する。

【０１１６】その場合、受信側に３つの問題が生じ得る。すなわち、受信側はサブエージェントＩＤについてのローカルエントリーを持たない場合があり；クラスＩＤについてのローカルエントリーを持たない可能性があり、インスタンスＩＤについてのローカルエントリーを持たない可能性がある。サブエージェントＩＤまたはクラスＩＤを持たない場合、受信側は送信側に対して要求を送り返して、欠落しているサブエージェントのグローバル識別記号またはクラスの絶対名称を尋ねることができる。グローバル識別記号を受け取ったら、受信側はサブエージェントについてのインスタンスをすでに持っているか（単に、受信側がそれについての送信側のＩＤを持っていなかった）、あるいは受信側がそのサブエージェントについて聞くのがそれが最初であって、新たなサブエージェントインスタンスを作るかのいずれかである。

【０１１７】受信側がそのインスタンスについてのローカルエントリーを持たない場合、それの次の段階はインスタンスのクラスによって決まる。それがサブエージェントである場合、受信側は上記の方法によって、グローバル識別記号を要求する。それがインターンされた文字列である場合、受信側は送信側に対して文字列中の文字を尋ね、ローカルコピーを使用するか、あるいはそれを作る。他の場合はいずれの場合も、受信側は、それ以上の通信を行わずに、インスタンスに対する遠隔参照符を作ることができる。

【０１１８】（４）遅延メッセージおよび保留インスタンス

上記のように、未知のサブエージェントまたはインターンされた文字列に対する参照符を有するメッセージを受け取ることができる。そのようなメッセージは、関連情報が送信側から到着するまで遅延させられる事が望まれる。同一の未知インスタンスを参照する他のメッセージが、情報についての要求が送信された後で、応答を受け取る前に到着する場合がある。それらのメッセージも遅延させることができる。

【０１１９】受信したが未知であるサブエージェントおよびインターンされた文字列についての情報は、サブエージェントインスタンス中の「保留」ベクトルに保存される。解析ベクトル中にＩＤが見つからなければ、保留ベクトル中でそれを探す。そこで見つかれば、そのインスタンスのデータについての要求はすでに送られており、現在のメッセージはその情報が到着するまで遅延させなければならない。

【０１２０】（ｉ）不要オブジェクト回収

エージェント内のオブジェクトはマシン集合間に分散していることから、グローバルで非同期式の不要オブジェ

クト回収戦略が望ましい。分散参照カウントの機構を用いて、あるインスタンスについて、それへの遠隔参照がすでに回収されているかどうかを識別ができるようにする事が望ましい。各グローバルＩＤは、非ゼロの参照カウントを持つ。インスタンスを別のサブエージェントに送る場合、上記の３つのＩＤとともに、１以上の参照カウントを送る必要がある。これらの参照カウントは、受信側サブエージェントにあるグローバルＩＤのカウントに加算される。

【０１２１】メッセージ中のインスタンスが、参照カウントが１であるグローバルＩＤを有する場合、送信側サブエージェントはメッセージ送信を遅延させなければならない。

【０１２２】参照カウントなしにインスタンスを送ることはできず、また、持っているカウントは１は保持しなければならない。この場合、現在そのインスタンスを持っているサブエージェントは、追加の参照カウントを要求する。それが到着すると、新たに到着した参照カウントの一部とともにメッセージを送ることが可能になる。あるグローバルＩＤがあるサブエージェントからそれ以上参照されなくなると、その参照カウントは、そのインスタンスを持つサブエージェントに送り返される。そのサブエージェントが、インスタンスに関する全ての現存カウントを受信したら、そのインスタンスはエージェントのローカル不要オブジェクト回収装置によって再利用することが可能となる。

本発明に関する用途例

当業者であれば容易に理解できるように、本発明の分散エージェントシステムは、分散計算の分野で広い利用可能性を有することは明らかであり、モデムによる低バンド幅・高待ち時間の通信から、高性能コンピュータ群で見られるような高バンド幅・低待ち時間の通信に至る広い範囲のネットワーク通信システム上で実現することができる。そのような用途の特定の例として、本発明は、移動性が重要となるネットワーク指向アプリケーションのための効果的な支援を提供する。そのような用途には、企業のイントラネットでのデータの検索および更新を自動的に行うことができる移動型ソフトウェアアシスタントや、問い合わせ処理や、ネットワーク中のマシン間でのデータベース状態の移動を行って、有効性やバンド幅を最適化する適応型のクエリーエンジンなどがある。さらに、データマイニング、保管業務、検索アプリケーションなどの高性能が要求される分散アプリケーションにも、本発明を用いることが有効であろう。上記の例は、単に例示的なものであって、本発明についての多数かつ多様な用途のうちのごく少数を示すだけであることは、理解しておくべき点である。

【０１２３】以上、プロセスおよびエージェントの状態を複数の異質であっても良いマシンに分散させることができ、別のマシンにあるデータの透明アクセスを可能とし、別個のマシン間で全体または部分的に、容易かつ有効なプロセス移動を可能とするオブジェクトに基づくカプセル化モデル（エージェント）を提供する分散エージェントシステムについて説明したが、本明細書に添付の特許請求の範囲によってのみ限定される本発明の広義の内容および精神から逸脱しない限りにおいて、さらなる変更・修正が可能であることは、当業者には明らかであろう。

【０１２４】

【発明の効果】以上説明したように、本発明には以下の効果がある。

【０１２５】本発明の各分散エージェントは、ネットワークのマシン中の１台、数台もしくは多数台間に分散されて、操作の同時性を高くすることができ、しかも同時に、エージェント内のタスクおよびデータ（それ自体、ネットワークのマシン間に分散されていてもよい）を、そのネットワークで動作する他のタスクおよびデータならびにそのようなタスクおよびデータが存在する同一マシンでのそのタスクおよびデータによる障害から保護する、保護されたカプセル化ソフトウェア構造を維持することができる。そのようなエージェントの移動は、プロセス実行中であっても簡単であり、ネットワークを通じて一貫性を保っている。具体的には、エージェント自体に対する事前の通知なしに移動させた後に、他のエージェントがある特定のエージェントに対して継続してアクセスすることができる。

【図面の簡単な説明】

【図１】本発明による分散エージェントシステムの基本的構成要素を示す模式図である。

【図２】本発明による分散エージェントシステムにおける、同一エージェント内のオブジェクト間での通信、ならびに異なるエージェントにおけるオブジェクト間での通信を示す模式図である。

【図３】本発明による分散エージェントシステムでの、オブジェクト空間の動作を示す概念図である。

【図４】（ａ）　本発明による分散エージェントシステムでの、標的ベースへの移動前のソースベース上のエージェントを示す模式図である。

（ｂ）　本発明による分散エージェントシステムでの、ソースベースから標的ベースへのエージェントの弱い移動を示す模式図である。

（ｃ）　本発明による分散エージェントシステムでの、ソースベースから標的ベースへのエージェントの完全移動を示す模式図である。

【図５】本発明による分散エージェントシステムでの、エージェントをサーバーが提供する特定ベースに移動させる要求を扱うサーバーを示す模式図である。

【図６】本発明のネットワーク中心的移動を実行する方法の例を示すフローチャートである。

【図７】本発明による分散エージェントシステムで、Ｒ

ＰＣモデルと呼び出し元モデルの両方の下でのメソッド呼び出しを説明する、２つの別個のベースで動作するエージェントの模式図である。

【図８】本発明による分散エージェントシステムでの、コアライブラリクラスにおけるオブジェクトの移動を示す模式図である。

【図９】本発明による分散エージェントシステムでの、ユーザー定義クラスにおけるオブジェクト移動を示す模式図である。

【図１０】本発明による分散エージェントシステムでの、新たなクラスオブジェクト作成を示す模式図である。

【図１１】実行時システムの、本発明による分散エージェントシステムの他の基本的構成要素に対する関係を示す模式図である。

【図１２】本発明による分散エージェントシステムにおけるベースのサブコンポーネント、サブエージェントおよび実行時システムを示す模式図である。

【図１３】本発明の実行時システムの、本発明による分散エージェントシステムにおけるエージェント移動の順序例との関係を示す模式図である。

【図１４】本発明の実行時システムの、本発明による分散エージェントシステムにおけるエージェント移動の順序例との関係を示す模式図である。

【図１５】本発明の実行時システムの、本発明による分散エージェントシステムにおけるエージェント移動の順序例との関係を示す模式図である。

【図１６】本発明の実行時システムの、本発明による分散エージェントシステムにおけるエージェント移動の順序例との関係を示す模式図である。

【図１７】本発明の実行時システムの、本発明による分散エージェントシステムにおけるエージェント移動の順序例との関係を示す模式図である。

【図１８】本発明の実行時システムと、本発明による分散エージェントシステムにおける部分エージェント移動の例との関係を示す模式図である。

【図１９】本発明の実行時システムと、本発明による分散エージェントシステムにおける部分エージェント移動の例との関係を示す模式図である。

【図２０】本発明の実行時システムと、本発明による分散エージェントシステムにおける全体エージェント移動の例との関係を示す模式図である。

【図２１】本発明の実行時システムと、本発明による分散エージェントシステムにおける全体エージェント移動の例との関係を示す模式図である。

【図２２】本発明の実行時システムと、本発明による分散エージェントシステムにおけるオブジェクト移動の例との関係を示す模式図である。

【図２３】本発明の実行時システムと、本発明による分散エージェントシステムにおけるオブジェクト移動の例との関係を示す模式図である。

【図２４】本発明による分散エージェントシステムにおけるエージェント移動の文脈での遠隔オブジェクトアクセスの第１の例を示す模式図である。

【図２５】本発明による分散エージェントシステムにおけるエージェント移動の文脈での遠隔オブジェクトアクセスの第１の例を示す模式図である。

【図２６】本発明による分散エージェントシステムにおけるエージェント移動の文脈での遠隔オブジェクトアクセスの第２の例を示す模式図である。

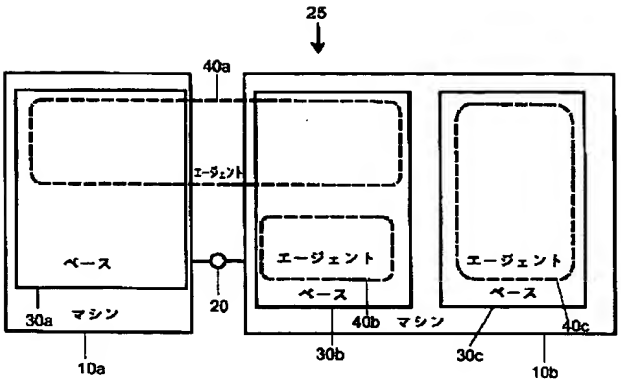
【図２７】本発明による分散エージェントシステムにおけるエージェント移動の文脈での遠隔オブジェクトアクセスの第２の例を示す模式図である。

【図２８】本発明による分散エージェントシステムで使用されるインスタンスオブジェクトおよびクラスオブジェクトを示す図である。

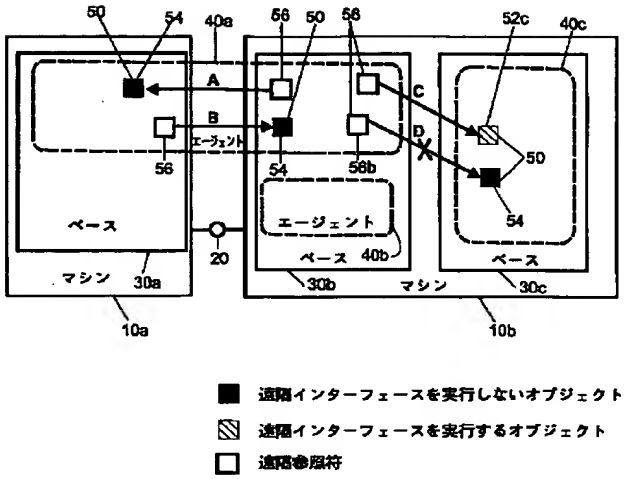
【符号の説明】

１０ａ、１０ｂ　マシン
２０　通信インターフェース
３０、３０ａ、３０ｂ、３０ｃ、３０ｊ　ベース
３０ｋ　標的ベース
４０ａ、４０ｂ、４０ｃ、４０ｊ、４０ｍ　エージェント
５０、５５　オブジェクト
５２ｃ　遠隔インターフェースを有するオブジェクト
５４、５４ｂ、５９　遠隔インターフェースを有しないオブジェクト
５５‘、５６　遠隔参照符
５７　移動可能なオブジェクト
５８　アンカー固定オブジェクト
６０　サーバー
７０　オブジェクト空間
２００　インスタンスブロック
２１０　クラスオブジェクト
２２０　メソッドコード

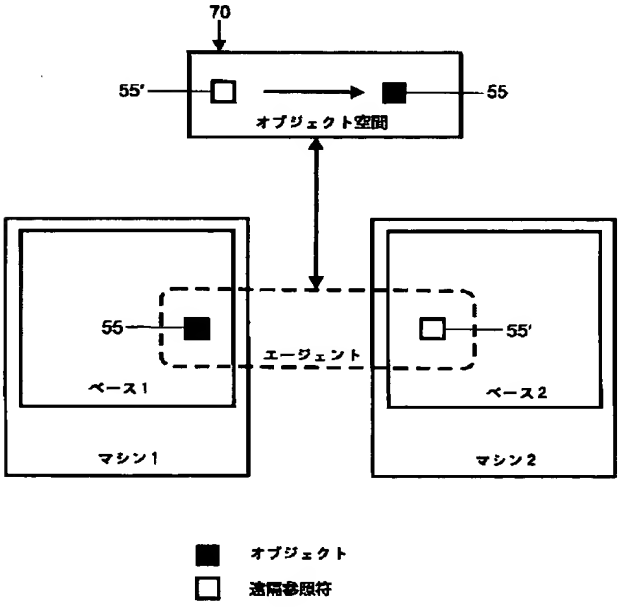
【図１】


25
40a
エージェント
ベース
30a マシン
10a
20
エージェント
ベース
40b
30b
マシン
エージェント
ベース
40c
30c
10b

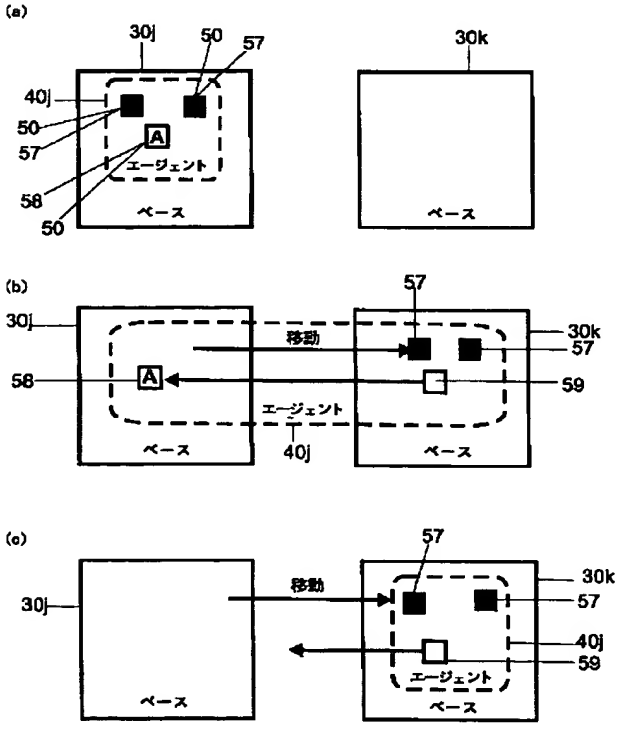
【図２】


50
54
40a
56
50
58
52c
40c
A
B
C
D
エージェント
56
54
58b
50
54
ベース
エージェント
20
ベース 40b
ベース
マシン 30a
30b
マシン
30c
10a
10b
遠隔インターフェースを実行しないオブジェクト
遠隔インターフェースを実行するオブジェクト
遠隔参照符

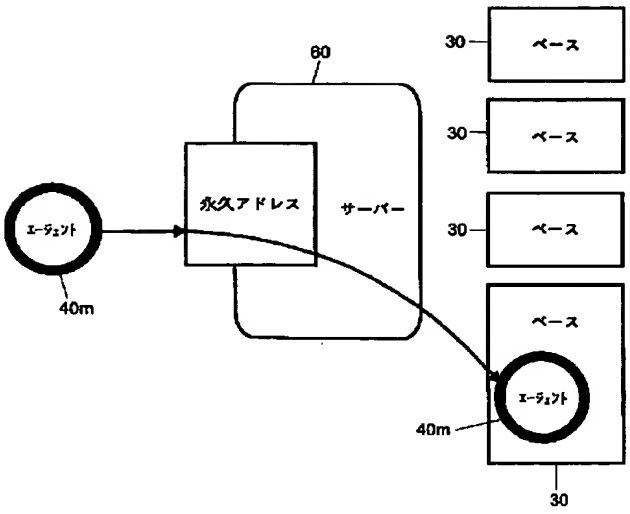
【図３】


70
55'
55
オブジェクト空間
55
エージェント
55'
ベース1
ベース2
マシン1
マシン2
オブジェクト
遠隔参照符

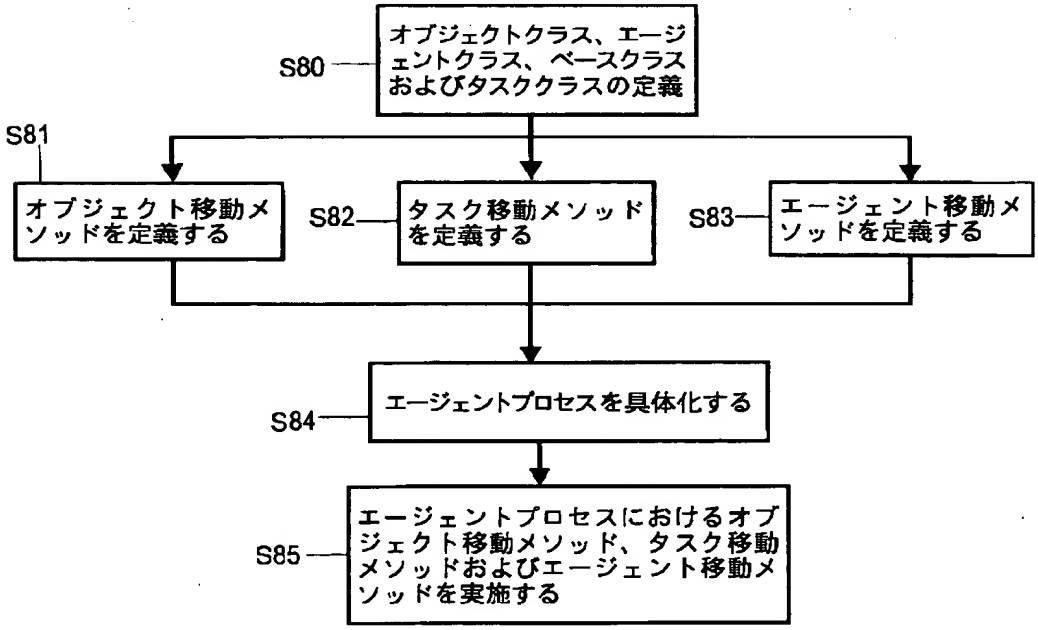
【図４】


(a)
30j
50
57
30k
40j
50
57
A
エージェント
58
50
ベース
ベース
(b)
57
30j
移動
30k
57
58
A
59
エージェント
ベース
40j
ベース
(c)
57
30j
移動
30k
57
40j
59
エージェント
ベース
ベース

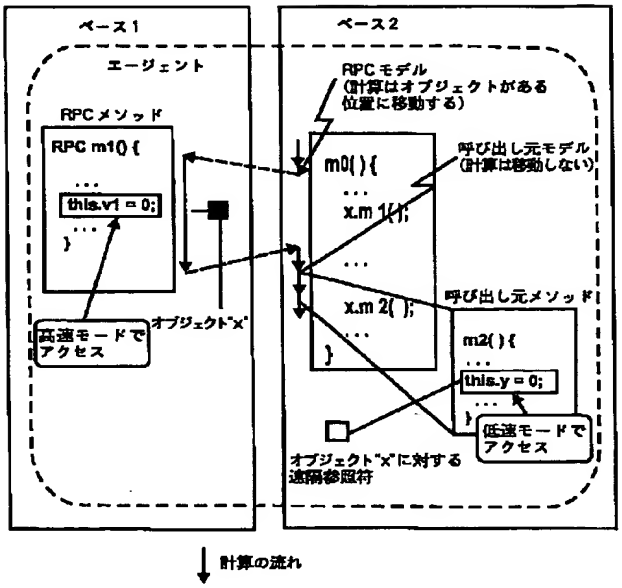
【図５】


60
30
ベース
30
ベース
30
ベース
ベース
永久アドレス
サーバー
エージェント
40m
エージェント
40m
30

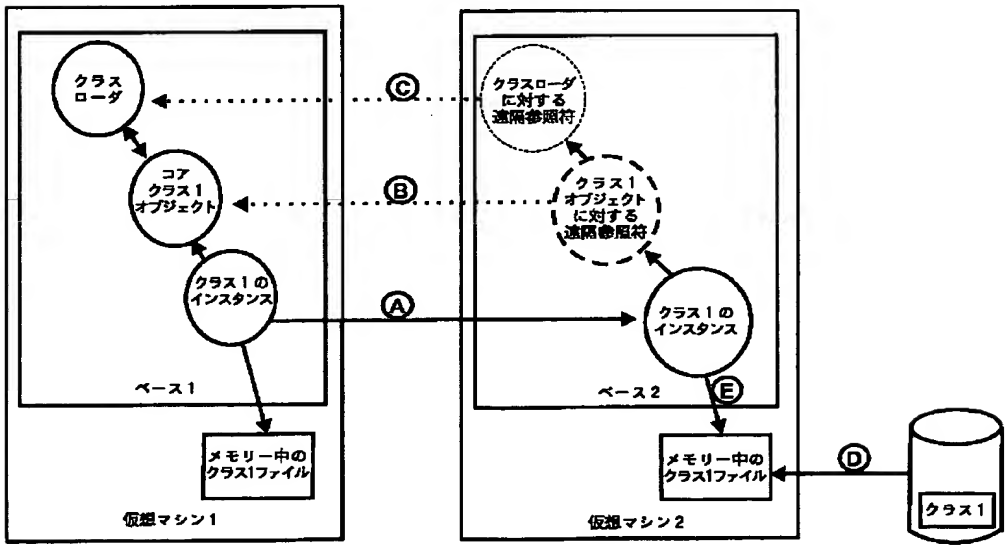
【図６】


S80
オブジェクトクラス、エージェントクラス、ベースクラスおよびタスククラスの定義
S81
オブジェクト移動メソッドを定義する
S82
タスク移動メソッドを定義する
S83
エージェント移動メソッドを定義する
S84
エージェントプロセスを具体化する
S85
エージェントプロセスにおけるオブジェクト移動メソッド、タスク移動メソッドおよびエージェント移動メソッドを実施する

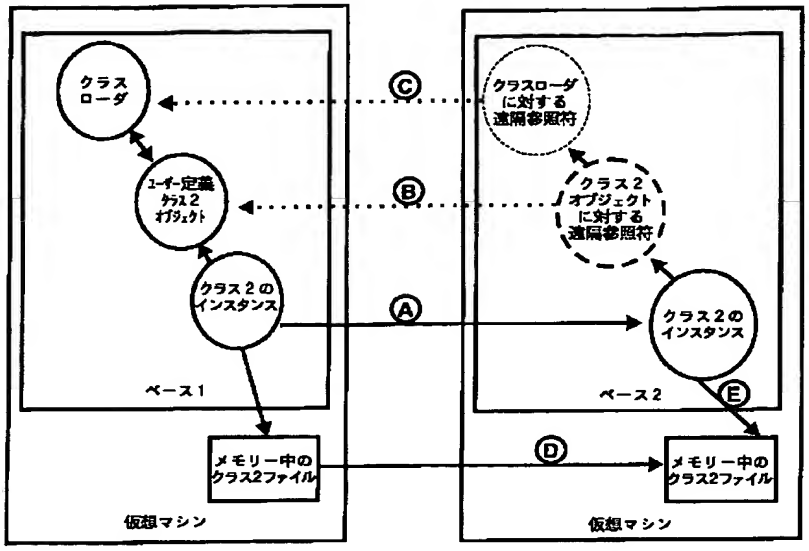
【図７】


ベース1
ベース2
エージェント
RPCモデル
(計算はオブジェクトがある
位置に移動する)
RPCメソッド
RPC m1() {
this.v1 = 0;
}
m0() {
x.m 1();
x.m 2( );
}
呼び出し元モデル
(計算は移動しない)
呼び出し元メソッド
m2() {
this.y = 0;
}
高速モードで
アクセス
オブジェクト"x"
低速モードで
アクセス
オブジェクト"x"に対する
遠隔参照符
計算の流れ

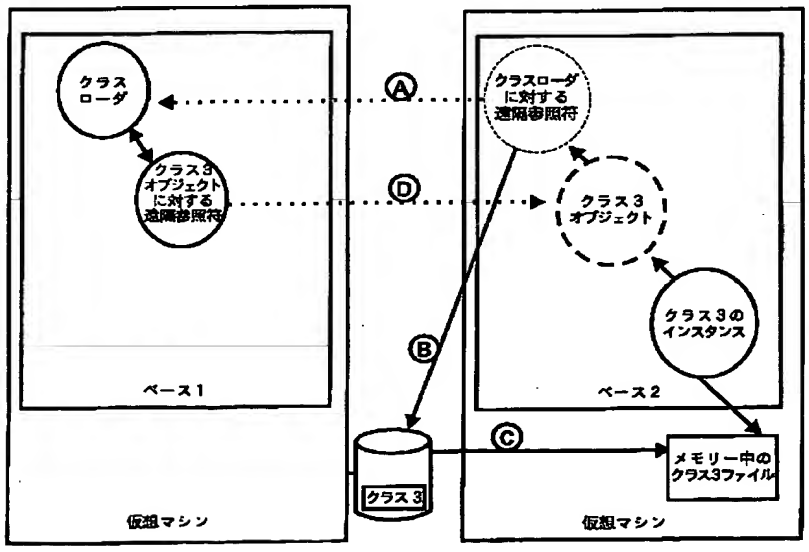
【図８】


クラス
ローダ
コア
クラス1
オブジェクト
クラス1の
インスタンス
ベース1
メモリー中の
クラス1ファイル
仮想マシン1
クラスローダ
に対する
遠隔参照符
クラス1
オブジェクト
に対する
遠隔参照符
クラス1の
インスタンス
ベース2
メモリー中の
クラス1ファイル
仮想マシン2
クラス1
A
B
C
D
E

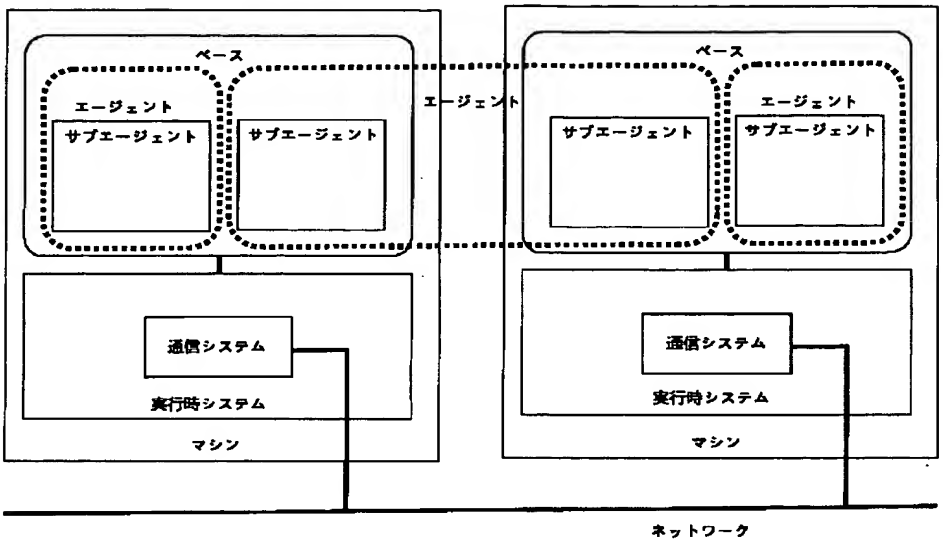
【図９】


クラス
ローダ
ユーザー定義
クラス２
オブジェクト
クラス２の
インスタンス
ベース１
メモリー中の
クラス2ファイル
仮想マシン
Ⓒ
Ⓑ
Ⓐ
Ⓓ
クラスローダ
に対する
遠隔参照符
クラス２
オブジェクト
に対する
遠隔参照符
クラス２の
インスタンス
ベース２
Ⓔ
メモリー中の
クラス2ファイル
仮想マシン

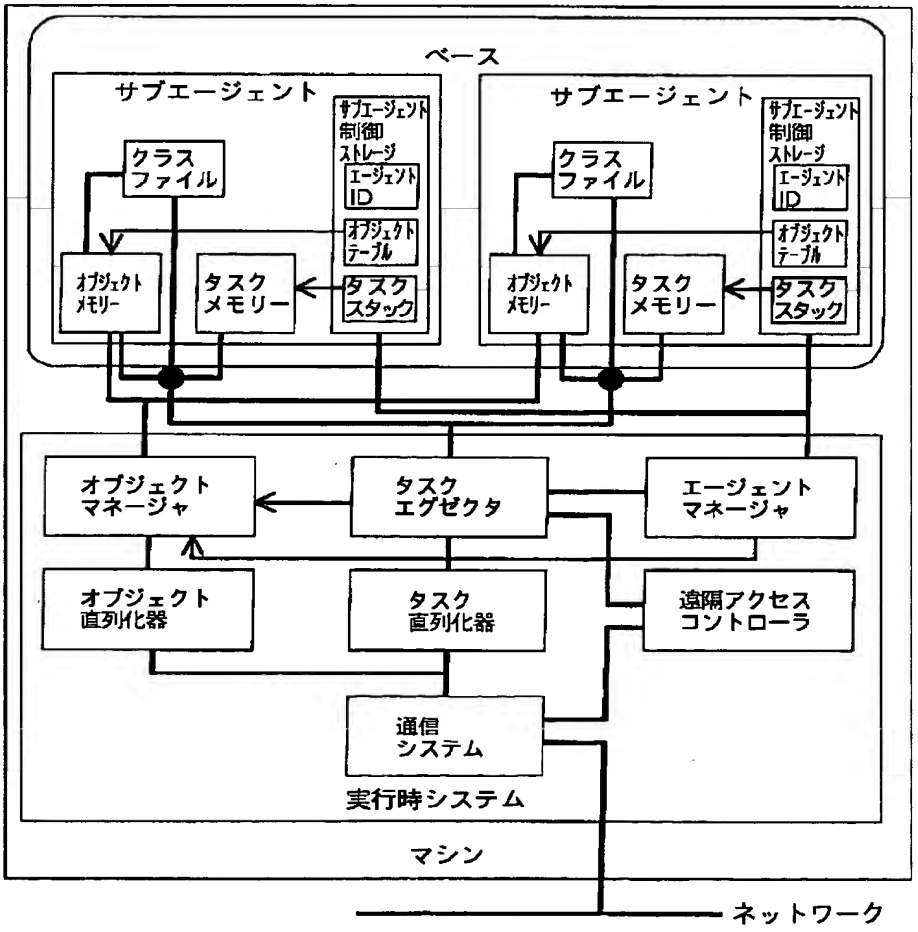
【図１０】


クラス
ローダ
クラス３
オブジェクト
に対する
遠隔参照符
ベース１
仮想マシン
Ⓐ
Ⓓ
Ⓑ
Ⓒ
クラス３
クラスローダ
に対する
遠隔参照符
クラス３
オブジェクト
クラス３の
インスタンス
ベース２
メモリー中の
クラス3ファイル
仮想マシン

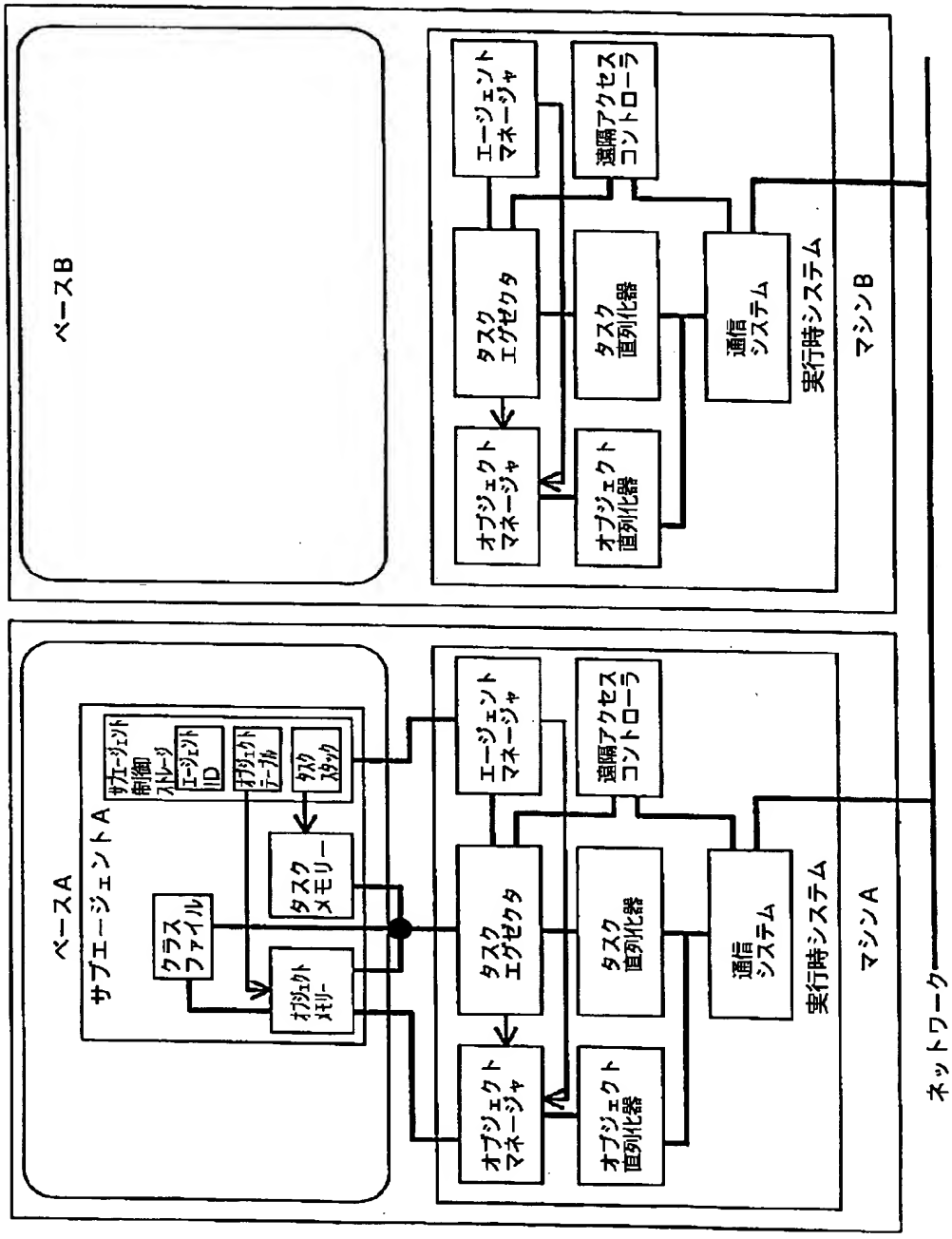
【図１１】


ベース
エージェント
サブエージェント
サブエージェント
エージェント
ベース
エージェント
サブエージェント
サブエージェント
通信システム
実行時システム
マシン
通信システム
実行時システム
マシン
ネットワーク

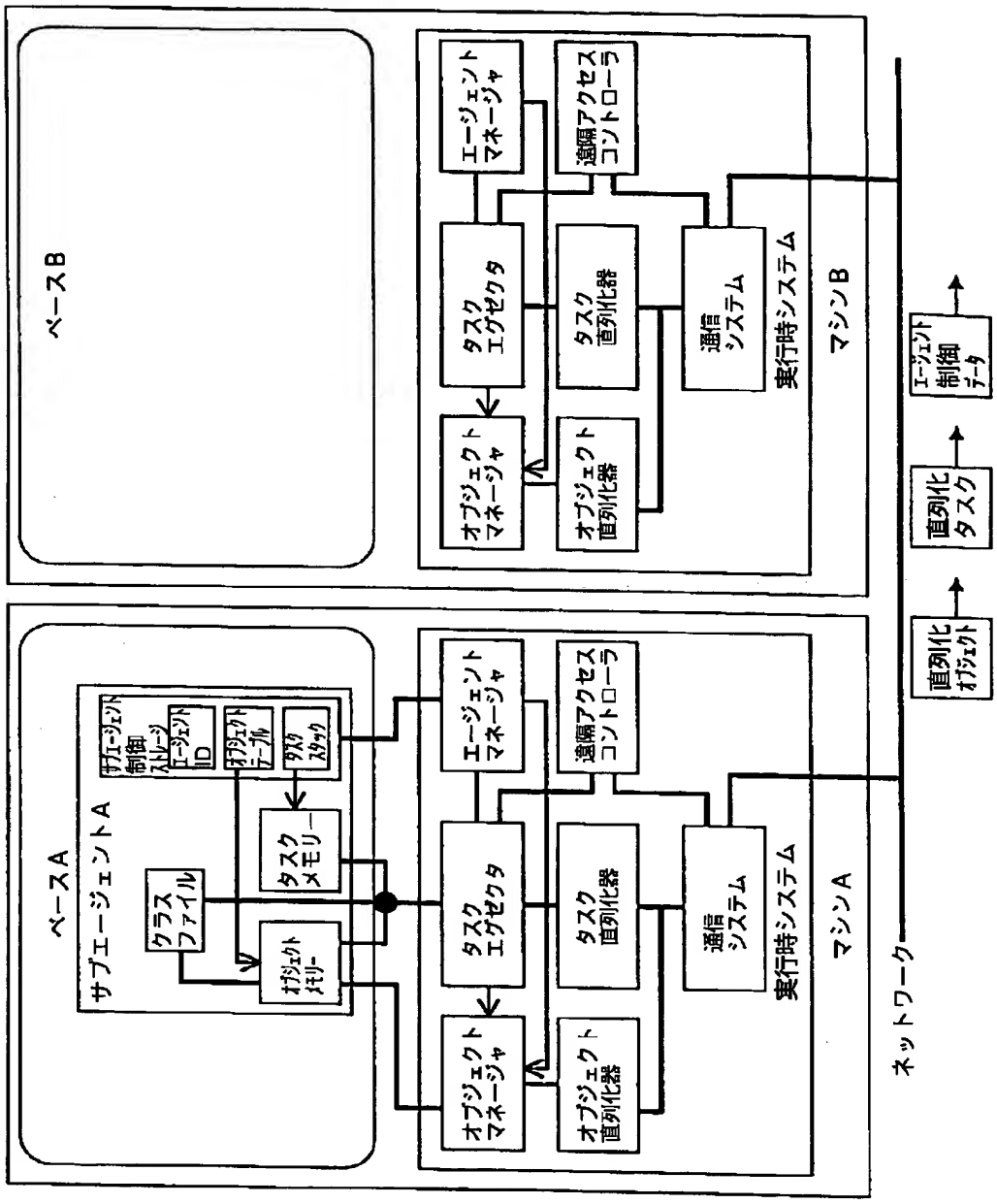
【図１２】


ベース
サブエージェント
サブエージェント制御ストレージ
エージェントID
オブジェクトテーブル
タスクスタック
クラスファイル
オブジェクトメモリー
タスクメモリー
サブエージェント
サブエージェント制御ストレージ
エージェントID
オブジェクトテーブル
タスクスタック
クラスファイル
オブジェクトメモリー
タスクメモリー
オブジェクトマネージャ
タスクエグゼクタ
エージェントマネージャ
オブジェクト直列化器
タスク直列化器
遠隔アクセスコントローラ
通信システム
実行時システム
マシン
ネットワーク

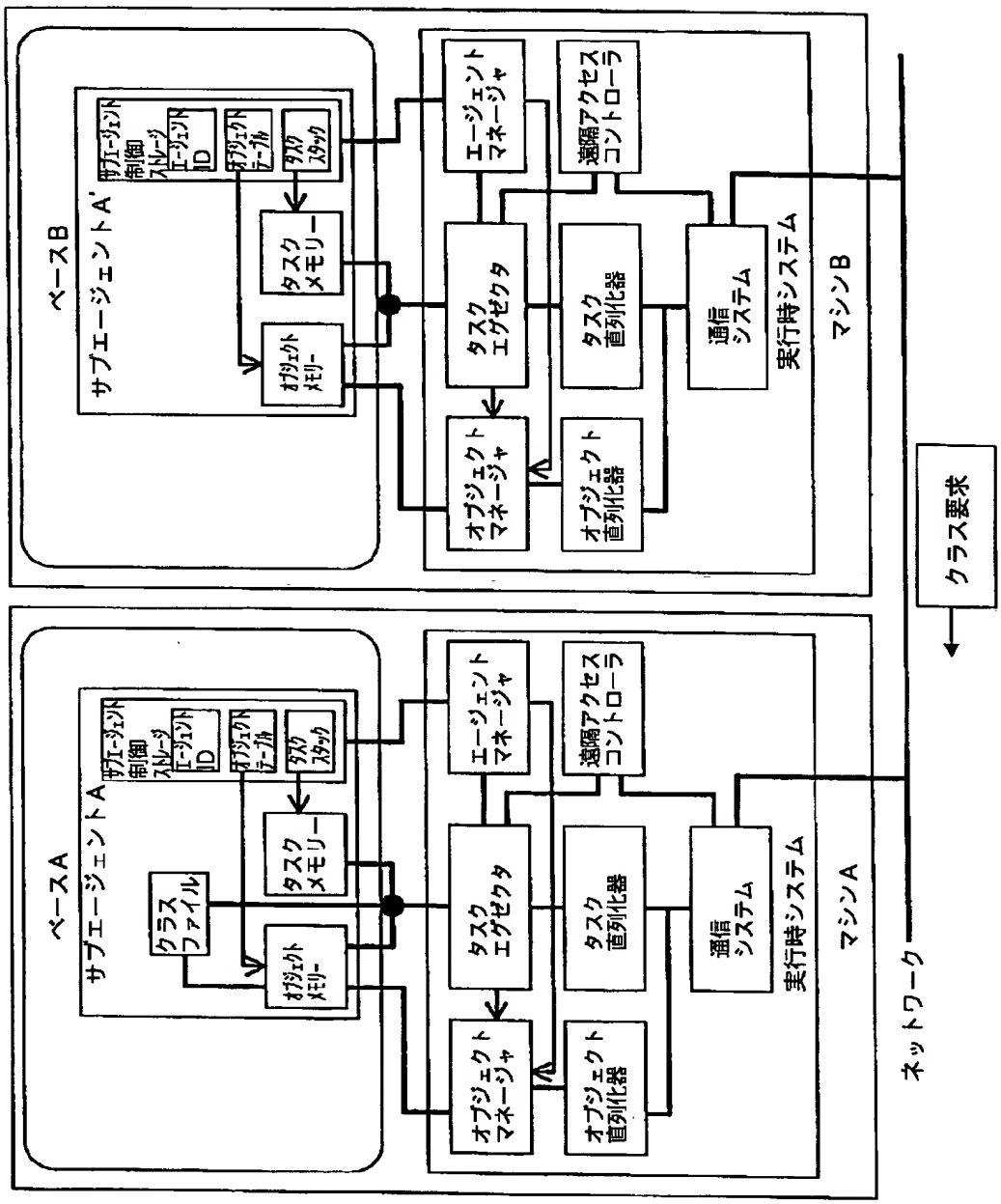
【図１３】


ベースB
エージェントマネージャ
遠隔アクセスコントローラ
タスクエグゼクタ
タスク直列化器
通信システム
オブジェクトマネージャ
オブジェクト直列化器
実行時システム
マシンB
ベースA
サブエージェントA
サブエージェント制御ストレージ
エージェントID
オブジェクトテーブル
タスクスタック
クラスファイル
タスクメモリー
オブジェクトメモリー
エージェントマネージャ
遠隔アクセスコントローラ
タスクエグゼクタ
タスク直列化器
通信システム
オブジェクトマネージャ
オブジェクト直列化器
実行時システム
マシンA
ネットワーク

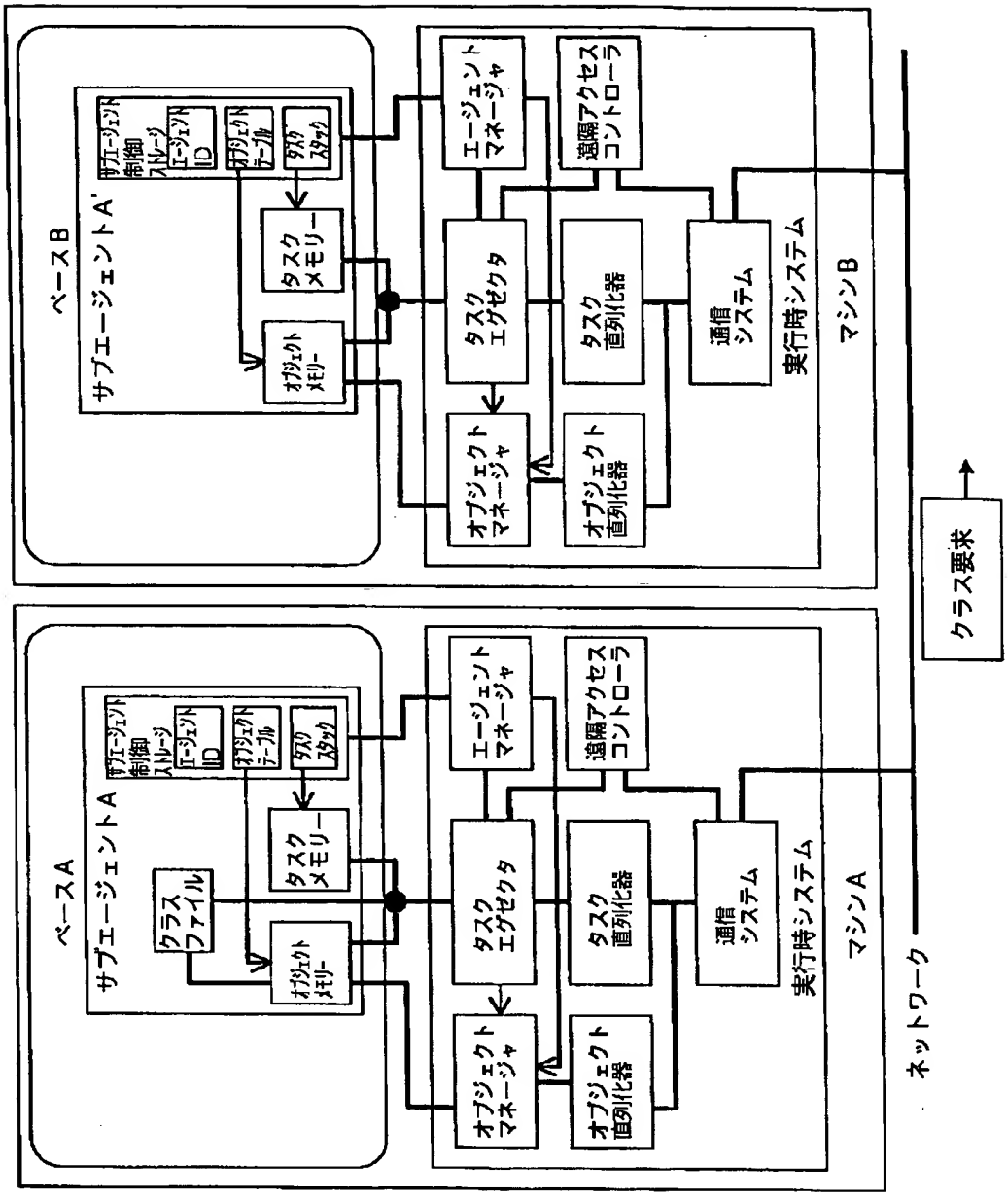
【図１４】


ベースB
エージェントマネージャ
遠隔アクセスコントローラ
タスクエグゼクタ
タスク直列化器
通信システム
オブジェクトマネージャ
オブジェクト直列化器
実行時システム
マシンB
エージェント制御データ
直列化タスク
直列化オブジェクト
ネットワーク
ベースA
サブエージェントA
サブエージェント制御ストレージ
エージェントID
オブジェクトテーブル
タスクスタック
クラスファイル
タスクメモリー
オブジェクトメモリー
エージェントマネージャ
遠隔アクセスコントローラ
タスクエグゼクタ
タスク直列化器
通信システム
オブジェクトマネージャ
オブジェクト直列化器
実行時システム
マシンA

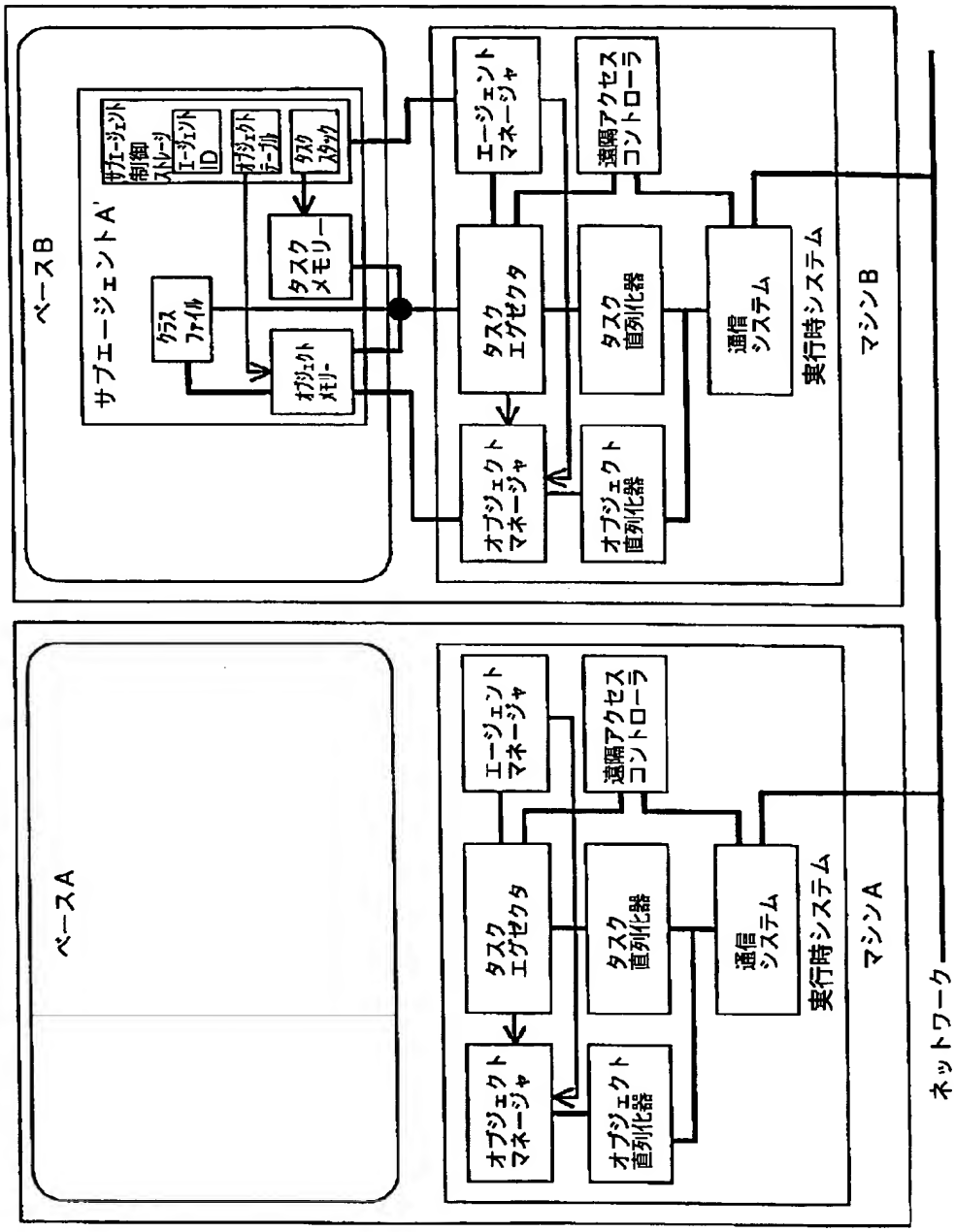
【図１５】


ベースA
サブエージェントA
サブエージェント制御ストレージ
エージェントID
オブジェクトテーブル
タスクスタック
クラスファイル
タスクメモリー
オブジェクトメモリー
エージェントマネージャ
遠隔アクセスコントローラ
タスクエグゼクタ
タスク直列化器
通信システム
オブジェクトマネージャ
オブジェクト直列化器
実行時システム
マシンA
ネットワーク
ベースB
サブエージェントA'
サブエージェント制御ストレージ
エージェントID
オブジェクトテーブル
タスクスタック
タスクメモリー
オブジェクトメモリー
エージェントマネージャ
遠隔アクセスコントローラ
タスクエグゼクタ
タスク直列化器
通信システム
オブジェクトマネージャ
オブジェクト直列化器
実行時システム
マシンB
クラス要求

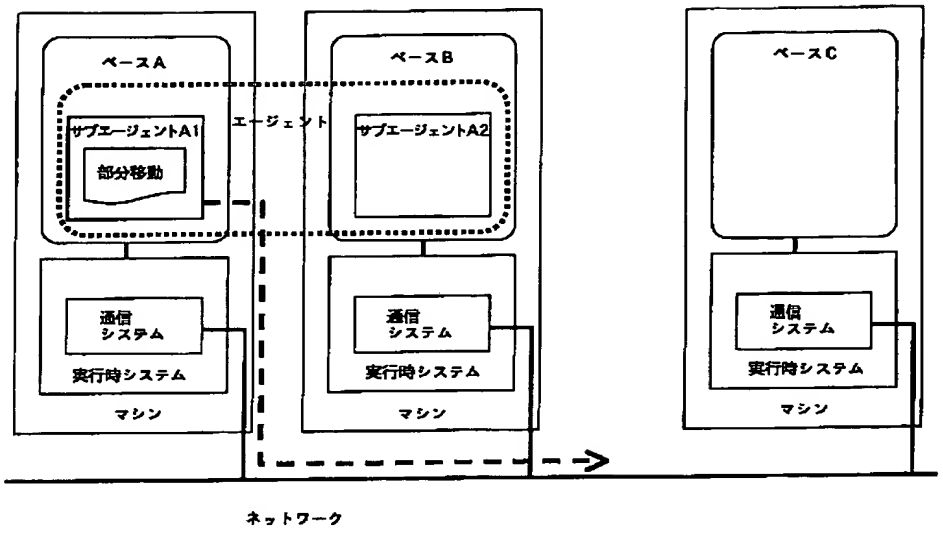
【図１６】


ベースB
サブエージェントA'
タスクメモリー
オブジェクトメモリー
エージェントマネージャ
遠隔アクセスコントローラ
タスクエグゼクタ
タスク直列化器
通信システム
オブジェクトマネージャ
オブジェクト直列化器
実行時システム
マシンB
ベースA
サブエージェントA
クラスファイル
マシンA
ネットワーク
クラス要求

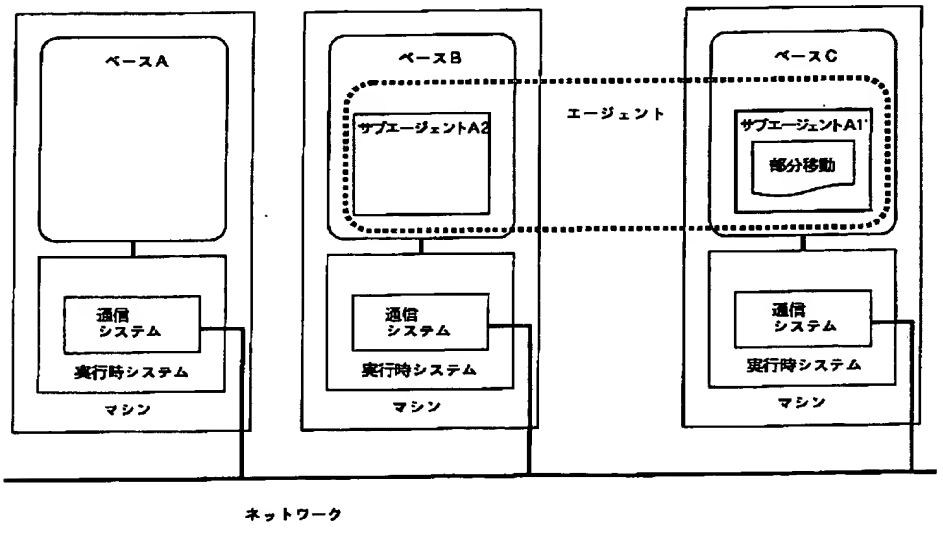
【図１７】


ベースA
ベースB
サブエージェントA'
サブエージェント制御ストレージ
エージェントID
オブジェクトモード
タスクスタック
タスクメモリー
クラスファイル
オブジェクトメモリー
エージェントマネージャ
遠隔アクセスコントローラ
タスクエグゼクタ
タスク直列化器
通信システム
オブジェクトマネージャ
オブジェクト直列化器
実行時システム
マシンA
マシンB
ネットワーク

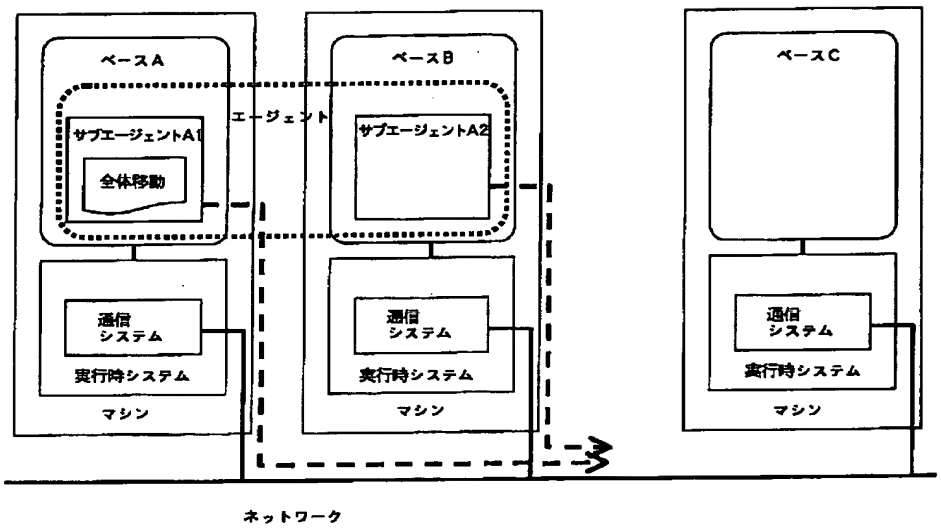
【図１８】


ベースA
ベースB
ベースC
エージェント
サブエージェントA1
部分移動
サブエージェントA2
通信システム
実行時システム
マシン
通信システム
実行時システム
マシン
通信システム
実行時システム
マシン
ネットワーク

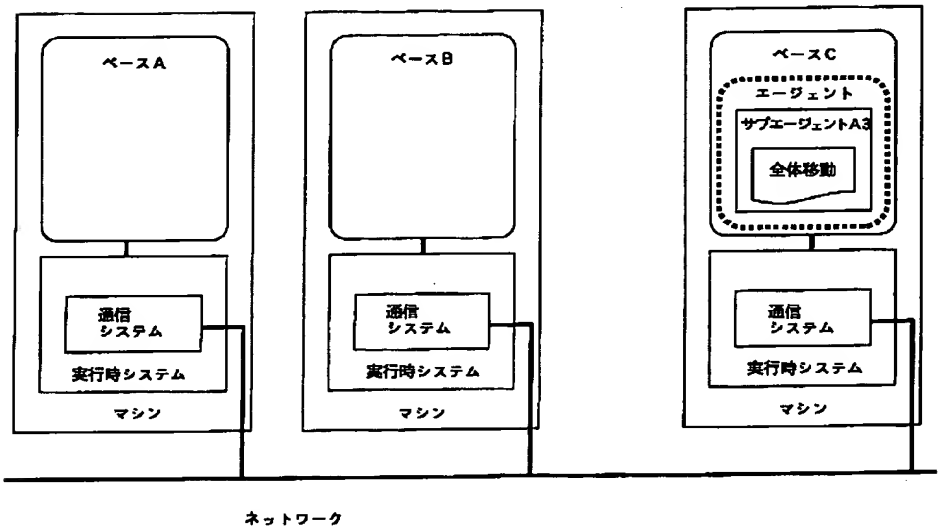
【図１９】


ベースA
ベースB
ベースC
エージェント
サブエージェントA2
サブエージェントA1'
部分移動
通信システム
実行時システム
マシン
通信システム
実行時システム
マシン
通信システム
実行時システム
マシン
ネットワーク

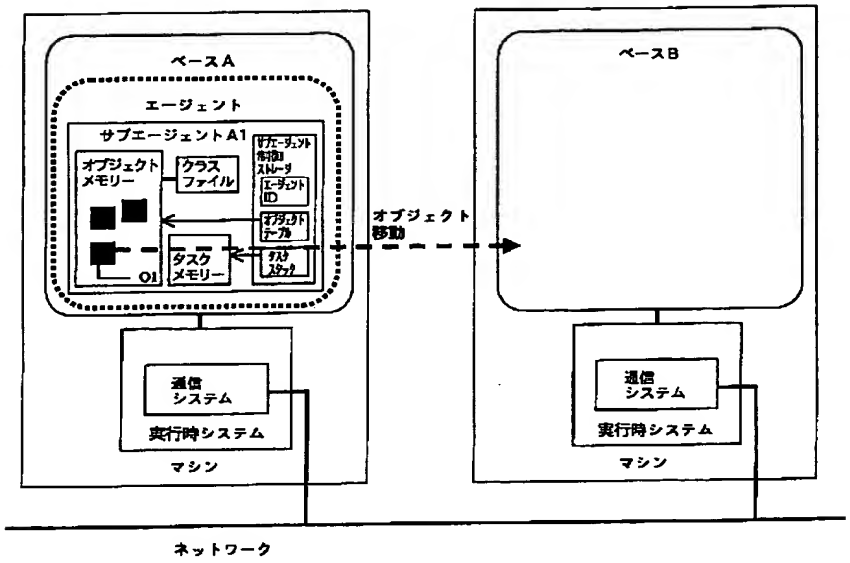
【図２０】


ベースA
ベースB
ベースC
エージェント
サブエージェントA1
全体移動
サブエージェントA2
通信システム
実行時システム
マシン
通信システム
実行時システム
マシン
通信システム
実行時システム
マシン
ネットワーク

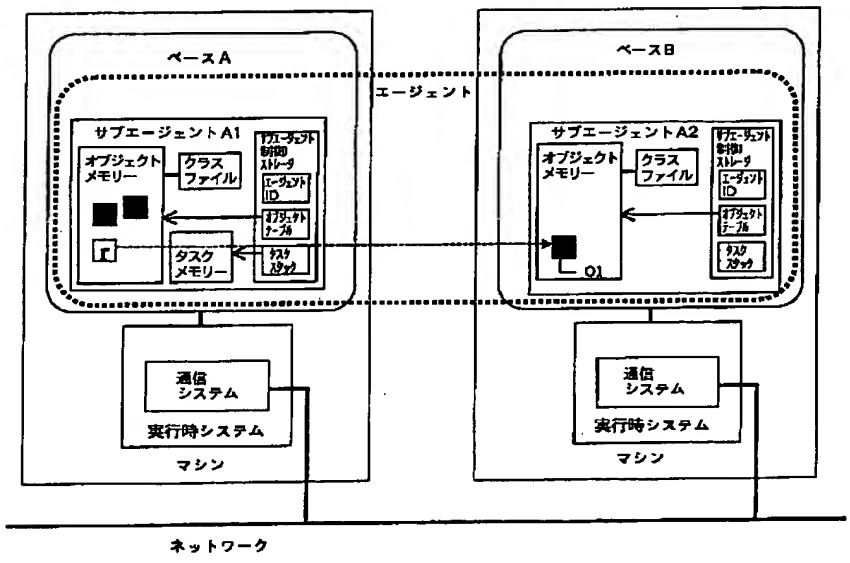
【図２１】


ベースA
ベースB
ベースC
エージェント
サブエージェントA3
全体移動
通信システム
実行時システム
マシン
通信システム
実行時システム
マシン
通信システム
実行時システム
マシン
ネットワーク

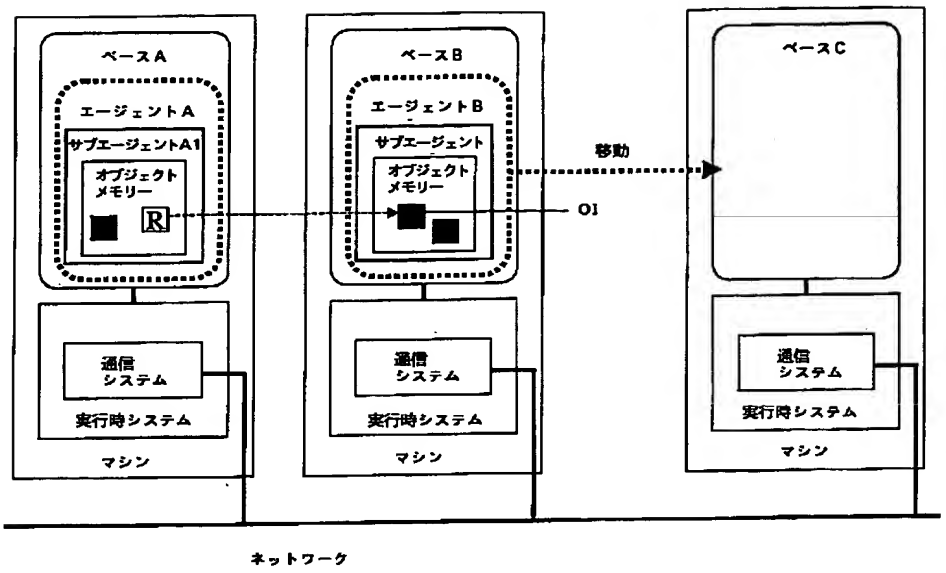
【図２２】


ベースA
エージェント
サブエージェントA1
オブジェクトメモリー
クラスファイル
エージェントID
オブジェクトテーブル
タスクスタック
タスクメモリー
O1
通信システム
実行時システム
マシン
オブジェクト移動
ベースB
通信システム
実行時システム
マシン
ネットワーク

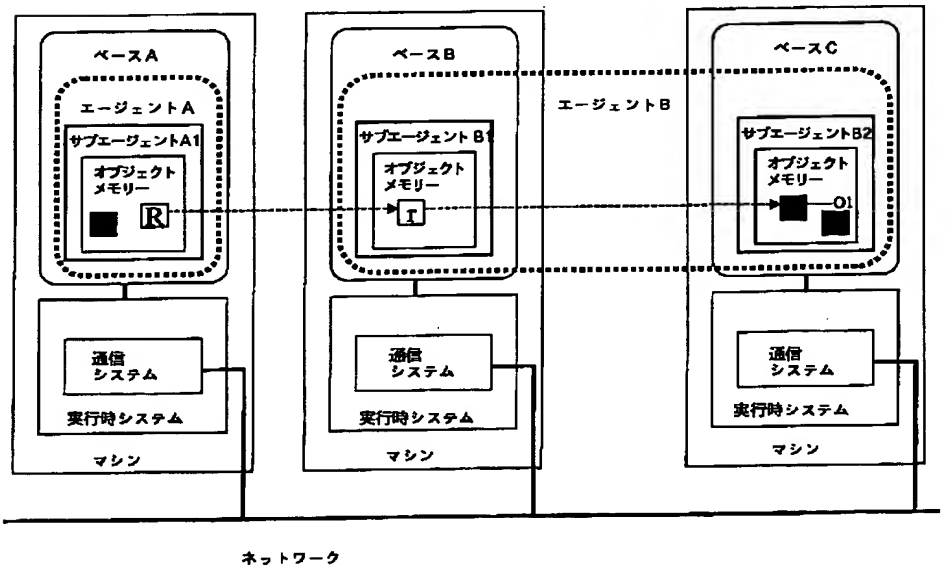
【図２３】


ベースA
エージェント
サブエージェントA1
オブジェクトメモリー
クラスファイル
エージェントID
オブジェクトテーブル
タスクスタック
タスクメモリー
通信システム
実行時システム
マシン
ベースB
サブエージェントA2
オブジェクトメモリー
クラスファイル
エージェントID
オブジェクトテーブル
タスクスタック
O1
通信システム
実行時システム
マシン
ネットワーク

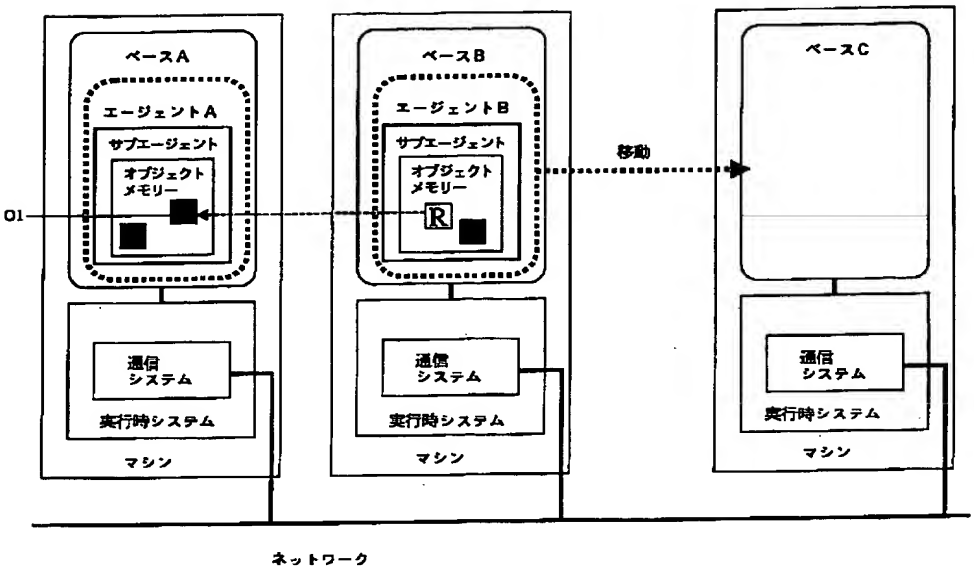
【図２４】


ベースA
エージェントA
サブエージェントA1
オブジェクト
メモリー
R
通信
システム
実行時システム
マシン
ベースB
エージェントB
サブエージェント
オブジェクト
メモリー
O1
通信
システム
実行時システム
マシン
移動
ベースC
通信
システム
実行時システム
マシン
ネットワーク

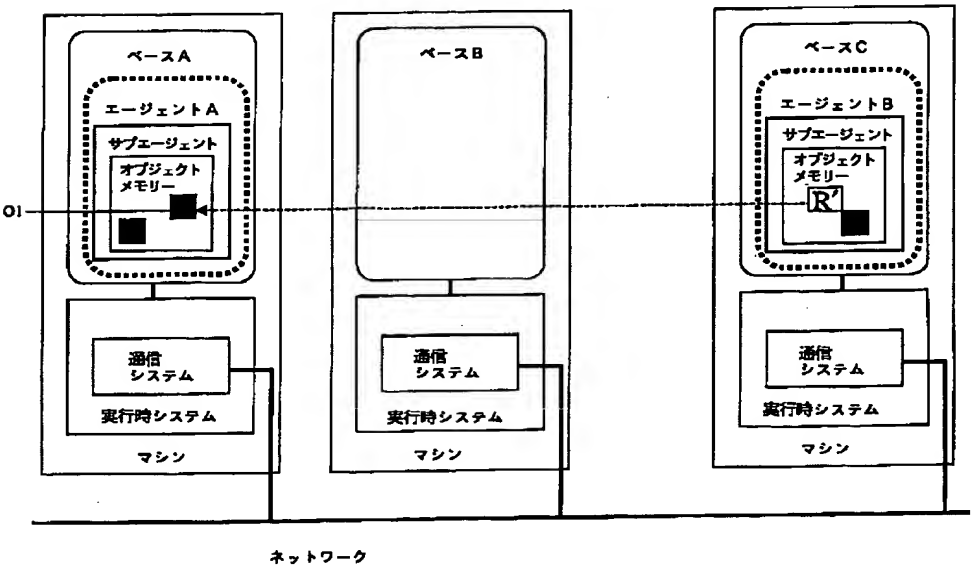
【図２５】


ベースA
エージェントA
サブエージェントA1
オブジェクト
メモリー
R
通信
システム
実行時システム
マシン
ベースB
エージェントB
サブエージェントB1
オブジェクト
メモリー
r
通信
システム
実行時システム
マシン
ベースC
サブエージェントB2
オブジェクト
メモリー
O1
通信
システム
実行時システム
マシン
ネットワーク

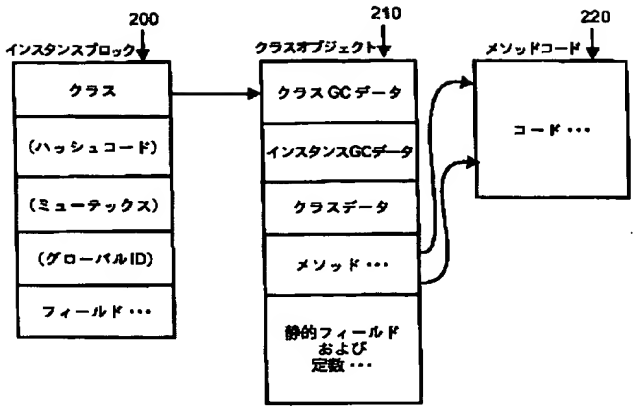
【図２６】


ベースA
エージェントA
サブエージェント
オブジェクト
メモリー
O1
通信
システム
実行時システム
マシン
ベースB
エージェントB
サブエージェント
オブジェクト
メモリー
R
通信
システム
実行時システム
マシン
移動
ベースC
通信
システム
実行時システム
マシン
ネットワーク

【図２７】


ベースA
エージェントA
サブエージェント
オブジェクト
メモリー
O1
通信
システム
実行時システム
マシン
ベースB
通信
システム
実行時システム
マシン
ベースC
エージェントB
サブエージェント
オブジェクト
メモリー
R'
通信
システム
実行時システム
マシン
ネットワーク

【図２８】



【手続補正書】
【提出日】平成１１年２月２５日（１９９９．２．２５）
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】発明の名称
【補正方法】変更
【補正内容】
【発明の名称】　ネットワークとして接続された複数のコンピュータマシン用の分散ソフトウェアシステム及びその実現手法

フロントページの続き

(72)発明者　リチャード　ケルシー
アメリカ合衆国、　ニュージャージー　08540、　プリンストン、　インディペンデンス　ウェイ　４　エヌ・イー・シー・リサーチ・インスティテューテュ・インク内
(72)発明者　ジェイムス　フィルビン
アメリカ合衆国、　ニュージャージー　08540、　プリンストン、　インディペンデンス　ウェイ　４　エヌ・イー・シー・リサーチ・インスティテューテュ・インク内
(72)発明者　藤田　悟
東京都港区芝五丁目７番１号　日本電気株式会社内
(72)発明者　小山　和也
東京都港区芝五丁目７番１号　日本電気株式会社内
(72)発明者　山之内　徹
東京都港区芝五丁目７番１号　日本電気株式会社内